Local Uniform Grid Refinement for Time-Dependent Partial Differential Equations

Local Uniform Grid Refinement for Time-Dependent Partial Differential Equations

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus,
prof. dr. P.W.M. de Meijer,
in het openbaar te verdedigen in de Aula der Universiteit
(Oude Lutherse Kerk, ingang Singel 411, hoek Spui).
op woensdag 26 januari 1994 te 13.30 uur

door

Ronald Alexander Trompert

geboren te Haarlem

Centrum voor Wiskunde en Informatica
Amsterdam
1994

Lokale uniforme roosterverfijning voor tijdsafhankelijke partiële differentiaalvergelijkingen

Promotor: Prof. Dr. P.J. van der Houwen

Co-promotor: Dr. J.G. Verwer

Faculteit: Wiskunde en Informatica

To Antoon and Hannie

Acknowledgements

First of all, I want to thank Dr. J.G. Verwer for his guidance and for giving me plenty of freedom to do research. His numerous contributions to this thesis are also indispensable. Further, I would like to express my gratitude to Prof.dr. P.J. van der Houwen for his willingness to be my promotor. I am grateful to Joke Blom and Paul Zegeling for the lively discussions we had about mathematics, Feyenoord and other subjects. Especially, I would like to thank Joke Blom for being a ‘wandelende manual’ to me, helping me out over and over ...... and over again with Unix and Ditroff etcetera, and I am also grateful for her "constructive" remarks about the subject of this thesis. Her criticism forced me to think things through more thoroughly and benefited this research this way.

Contents

Chapter 1

Introduction

Numerous processes in nature are described by models containing partial differential equations (PDEs). One can simulate these processes by solving the associated PDEs. The complexity of these PDEs, however, often demands that these equations are solved numerically. To obtain a numerical solution, one discretizes the PDEs on a set of discrete points, i.e. a space(-time) grid, which leads to a system of algebraic equations. Solving this system gives the approximate solution to the PDEs at these discrete nodes. This needs to be done only once when a PDE does not depend on time. When a PDE is time dependent then, starting from a known initial solution, this system of equations is subsequently solved in order to advance the approximate solution in time with a relatively small increment until a final time is reached. This procedure is called time stepping.

Many PDEs have solutions which are rapidly varying functions of the spatial or temporal co-ordinates. PDEs arising from models describing shock hydrodynamics, transport in porous media, combustion processes and plasma physics and so on, can serve as an example in this respect. For the numerical practice it is important to realize that the accuracy of the approximate solution depends on the variation of the true solution from node to node. The larger the variations, the finer a grid needs to be to obtain accurate results. In many applications, this dependency leads to huge computational costs.

In case a uniform space grid is used, which means that all grid cells are identical, the following situation can occur. Suppose that the variations over the spatial domain of the solution are only locally large and small anywhere else. In that case a uniform grid has to be fine to get an accurate approximation in the region of large variation. However, such a grid is then also fine in regions where such a high resolution is not really needed. The picture on the left of Figure 1.1 clearly shows this. This approach is computationally inefficient because the solution has to be computed at a very large number of nodes. Moreover, storage for all these values has to be provided too. Adaptive grid methods prove to be greatly beneficial here, since these methods attempt to obtain accurate results in such a situation with minimal computational effort, meaning CPU time and memory requirements. Adaptive methods do this by using a fine grid spacing only where it is really needed, i.e. where the large variations occur, and therefore, use as few nodes as possible. An example of an adaptive grid is shown in the picture on the right of Figure 1.1.

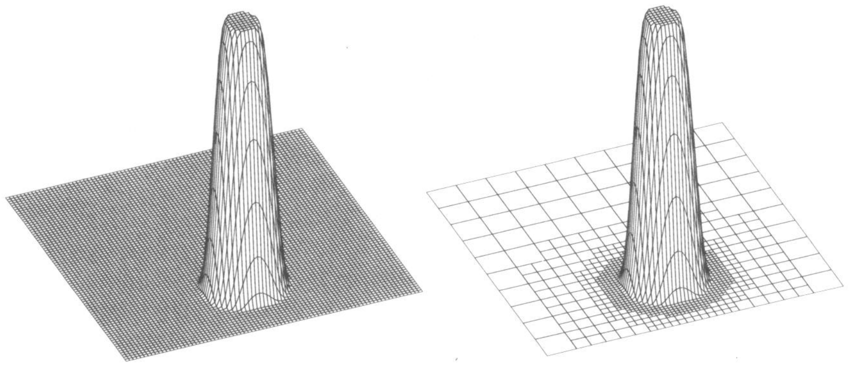


FIGURE 1.1. The solution computed with a uniform grid (left) and an adaptive grid (right).

Over the years a large number of adaptive grid methods have been proposed for time-dependent problems. Two main categories of adaptive grid methods can be distinguished, namely, *moving-grid* or *dynamic-regridding* methods and *static-regridding* methods.

In dynamic-regridding methods, nodes are moving continuously in the space-time domain, like in classical Lagrangian methods, and the discretization of the PDEs is coupled with the motion of the grid. Therefore, grids generated by moving-grid methods are essentially nonuniform and the number of nodes contained by such a grid is constant in time. Such methods can be found in [2, 9, 11, 13, 16, 21-23, 25, 28]. The motion of the grid can be governed in a number of different ways. There are methods were a system of ordinary differential equations (ODEs) describing the grid motion is solved together with the discretized PDEs. The moving-finite-element method of *Miller* et al [9, 21, 22], for example, is such a method. The grid movement in this method is generated by a least-squares minimization of the finite-element residual of the PDE over not only the nodal values of the solution and its time derivative, like in the standard Galerkin method, but also over the spatial co-ordinates of the nodes and their velocity. Other methods, like the one by *Dorfi* and *O'Drury* [11], use the equidistribution principle to generate this system of ODEs for the space grid movement. Equidistribution means that the method moves the nodes such that the nodal value of a non-negative monitor function multiplied with the corresponding cell size is constant over the whole grid. The method developed by *Petzold* [23] also generates a system of ODEs for the motion of the nodes. Here, the nodes are moved such that the rate of change of the solution in time is minimized. *Arney* and *Flaherty* [2] proposed a method which uses ODEs to compute the movement of a center of error of a cluster of nodes where the error is too large.

From this, the movement of each individual node is computed using a movement function. Further, there are also more heuristic approaches to move the grid, like in the method described by *Smooke* and *Koszykowski* [25], where the location of a node is extrapolated from its location at previous time points.

In static-regridding methods the location of nodes is fixed. A method of this type adapts the grid by adding nodes where they are necessary and removing them when they are no longer needed. The refinement or de-refinement is controlled by error estimates or error monitor values. Error monitors have no resemblance with the true numerical error. They are, for example, based on the slope or the curvature of the solution. Examples of static-regridding methods are described in [1,3,5-8,12,13,19,20]. Methods of this type are, for example, methods which embed new nodes in the existing grid. After refinement, the solution is then computed on a single nonuniform grid, even when the initial grid is uniform. Such methods are, for example, developed by *Adjerid* and *Flaherty* [1] and by *Bieterman* and *Babuška* [7]. Further, there is the adaptive method developed by *Maubach* [20] which treats the time-dependent problem like a boundary value problem. *Deuflhard* and his co-workers developed a finite-element adaptive multilevel method for parabolic PDEs [8,19] based on Rothe's method, which is also called the method of discretization in time first [24].

Another method of the static-regridding type is the local uniform grid refinement method, described, for example, in [3,5,6,12]. This method creates a series of increasingly finer local uniform subgrids where they are needed. The PDEs are solved at each grid separately for one time step in a consecutive order, from coarse to fine. These finer grids are not embedded in the coarser grid but are overlaying it. This method, which is the subject of this thesis, will be discussed in greater detail in the next section.

There are also methods which combine static with dynamic regridding. For example, *Gropp* [14] and *Arney* and *Flaherty* [4], have developed methods which combine local uniform grid refinement with dynamic regridding.

The difference between static and dynamic regridding is clearly illustrated by Figure 1.2. Here a local uniform grid refinement method described in [26] and the so-called moving-finite-element method [9,21,22,28] are applied to a combustion problem. The moving-finite-element pictures in this figure were obtained from [28]. This combustion problem is discussed in [26,28] and is derived from [1,18]. At the lower left corner of a square domain a front develops and propagates towards the upper and right boundary of the domain as time proceeds. Figure 1.2 shows this for both methods at two different time points.

The main advantage of static-regridding methods over moving-grid methods is that static-regridding methods are more robust than moving-grid methods. The phenomenon 'node crossing' in one space dimension and its equivalent 'grid distortion' in two or three space dimensions is a real danger for moving-grid methods and can reduce the accuracy of the computations considerably. In order to overcome this difficulty, user-defined parameters associated with penalty functions have to be introduced which limits the motion of the nodes. The choice of these parameters is left to the user and may be critical. However, not all moving-grid methods suffer

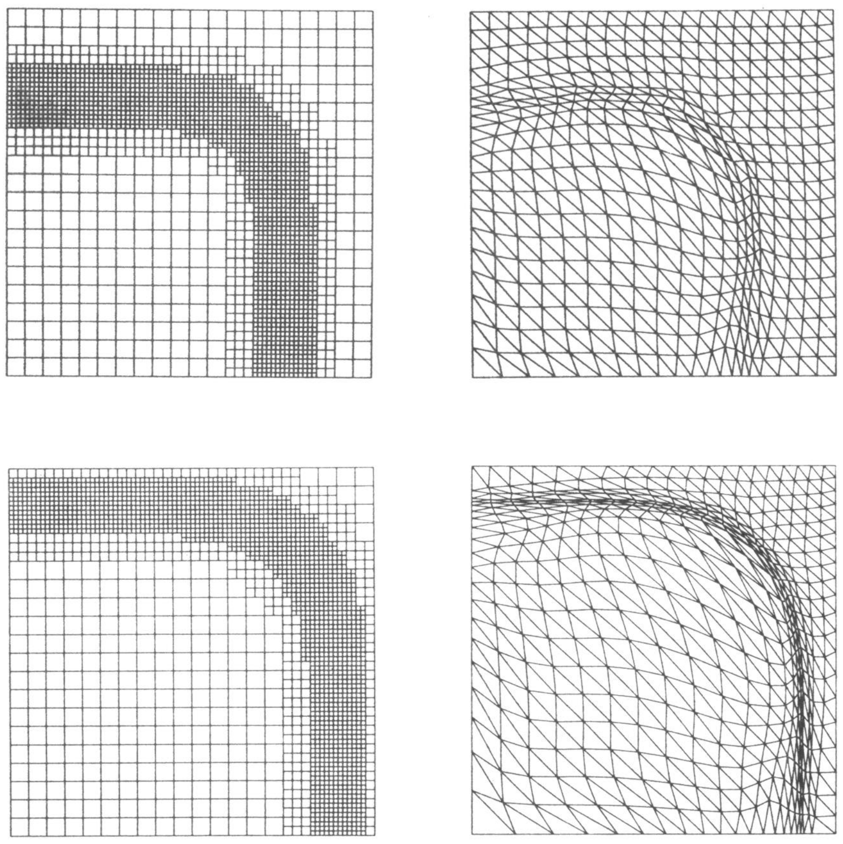


FIGURE 1.2. The grid associated with a static-regridding method (left) and a dynamic-regridding method (right) at two time points.

from this difficulty to the same degree. For example, the method in [2], does not have a serious grid-distortion problem, because the movement function of the nodes prevents this. It does, however, need a parameter to stabilize the motion of the grid. This brings us to more difficulties associated with moving-grid methods, namely, that grid motion based on minimization or equidistribution can be unstable [10] or may even be discontinuous in time [27] which complicates time stepping. Further, there are moving-grid methods where the nodes do not always move in the direction the user wants them to move and therefore, some a posteriori regridding to correct this deficiency in the moving-grid procedure is needed [23, 29].

Static-regridding methods do not have problems of this nature. Moreover, they only need a few user-defined parameters like, for example, error tolerances. In general, the choice of such parameters is not critical. However, moving-grid methods, in contrast to static-regridding methods, attempt to smooth the variations of the solution in the time direction, which allows larger time steps when the grid motion is sufficiently smooth. Finally, moving-grid methods, when working properly, use less nodes than static-regridding methods for a given accuracy.

In the next section, the subject of this thesis, which is the local uniform grid refinement method will be discussed.

1.1. The local uniform grid refinement method

The idea behind local uniform grid refinement is simple. Starting from a coarse base grid covering the whole domain, finer-and-finer uniform subgrids are recursively created locally in a nested manner in regions where the variations are large, or in other words, where the solution is steep. Each time step a new initial boundary value problem is solved at each grid separately in a consecutive order, from coarse to fine. Therefore, the local subgrids are not patched into the coarser grids but are actually overlaying them. An example of this is shown in Figure 1.3. Here we see the composite grid together with the uniform grids it consists of.

A finer grid uses a time step which is smaller or equal to the coarser-grid time step. In the latter case, the coarser-grid time step size is a whole multiple of the finer-grid time step size.

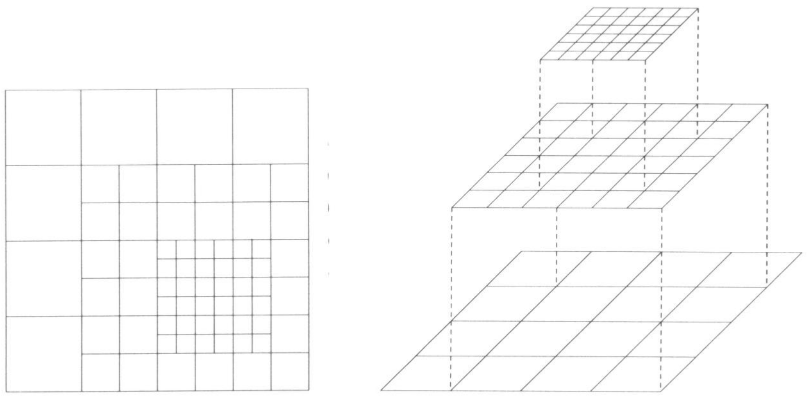


FIGURE 1.3. The local subgrids overlaying the coarser grid.

When a grid of a certain level of refinement has reached the same time level as a coarser grid, then the solution at the coarser grid is, in some way, updated by the solution at the finer grid. The location, shape and size of these subgrids are adjusted at discrete times to follow the movement of the steep parts of the solution. The generation of subgrids is continued until sufficient accuracy is reached at the finest subgrid. The subgrids in this method are properly nested, meaning that subgrids are completely overlapped by coarser grids.

So far, local uniform grid refinement methods have been proposed in a number of different varieties, applied to different kinds of PDEs. We will now sketch some varieties of the local uniform grid refinement method very briefly. The methods contained in [3, 4, 6, 14] are applied to hyperbolic PDEs and use explicit time stepping techniques. In all these methods, the subgrids having the same cell size can overlap each other. The method proposed by *Berger* and *Oliger* [6] employs rectangular subgrids which may be skewed with respect to the co-ordinate axes in order to align with the steep region of the solution. *Arney* and *Flaherty* [3] developed a method very similar to the one in [6] except that the subgrids here are created by so-called cellular refinement, meaning that the fine grid cells are properly nested within coarser grid cells. These subgrids have a piecewise polygonal shape. *Berger* and *Colella* [5] proposed a method comparable to the one in [6], except that the subgrids are now rectangles with sides parallel to the co-ordinate axes.

Local uniform grid refinement is combined with grid movement in [4, 14]. *Gropp* [4] proposed a method using subgrids which are rectangles having sides parallel to the co-ordinate axes and which are able to move as a whole with the moving steep fronts. In this method the subgrids are also allowed to overlap. In [14], *Arney* and *Flaherty* added grid movement to their method discussed in [3]. The nodes of the coarsest grid are able to move and the fine grid movement is induced by the movement of the coarsest grid. The grid movement technique used here is the same as in [2].

Local uniform grid refinement methods are used to solve elliptic PDES by *Gropp* and *Keyes* [15] and parabolic PDEs by *Flaherty*, *Moore* and *Ozturan* [12]. The subgrids in [12] are piecewise polygonal and the ones in [15] are rectangles. In both [12] and [15] domain decomposition is applied to improve the performance on parallel computers.

1.2. Contents of this thesis

This thesis is based on five papers which have appeared or are going to appear in the literature. Each chapter following this one contains one of these papers. The Chapters 2 and 6 are rather applied in nature while the Chapters 3,4 and 5 are more fundamental. The latter chapters form the kernel of this thesis. In these chapters strategies for grid refinement based on an error estimates are developed from error analyses. In general, a refinement strategy based on heuristic criteria like the slope or the curvature of the solution is computationally cheaper than an error-estimate-based strategy. However, a strategy based on error estimates will in many cases give more accurate results than a strategy based on heuristic error monitors. This is due to the fact that heuristic error monitors bear no relationship with the true numerical error. Because of this, the situation can occur that a strategy based on heuristics does not refine a grid cell which ought to be refined considering the numerical error and vice versa. Hence, a strategy based on error estimates can generate better subgrids than a heuristic strategy. A brief description of the contents of this thesis is given below.

Chapter 2 discusses the local uniform grid refinement method applied to parabolic PDEs. The explicit Runge-Kutta-Chebyshev (RKC) method of *van der Houwen* and *Sommeijer* [17] is used for time stepping, and the grid refinement process and time step selection are based on heuristic error monitors.

In Chapter 3 a refinement strategy, controlling the generation of subgrids, is developed based on an error analysis. The error analysis is carried out for the local uniform grid refinement method applied to time-dependent PDEs which after spatial discretization yield a system of ODEs, and where the implicit Euler method is used for time stepping. Further, it is assumed here that the grid spacing of the finest subgrid is fixed in time.

Chapter 4 is in many respects similar to Chapter 3, except that a general Runge-Kutta scheme is used for time stepping. The case of a variable grid spacing in time of the finest subgrid is also discussed in this chapter,

In Chapter 5, a refinement strategy is developed in case PDEs are solved which after spatial discretization result in a system of differential algebraic equations (DAEs). This refinement strategy is based on an error analysis carried out for this case. A backward differentiation formula method (BDF) is used for time stepping.

Chapter 6 discusses the application of the local uniform grid refinement method to a model for unsteady groundwater flow coupled with transport of solute in heterogeneous porous media. The local uniform grid refinement method proves to be useful for this application, since it frequently occurs that the variations of the concentration of solute over the spatial domain are only large at a small part of this domain. This work is carried out as a part of contract research in behalf of the RIVM - the Dutch National Institute of Public Health and Environmental Protection. In the scope of this project, the local uniform grid refinement method is implemented in a code called MOORKOP. This code is developed for solving a rather general class of PDEs defined on a rectangular domain, including transport problems in heterogeneous porous media.

REFERENCES

1. S. ADJERID and J.E. FLAHERTY (1988). A local Refinement Finite Element Method for Two Dimensional Parabolic Systems, *SIAM J. Sci. Statist. Comput.*, 9, 792-811.
2. D.C. ARNEY and J.E. FLAHERTY (1986). A Two-Dimensional Mesh-Moving Technique for Time-Dependent Partial Differential Equations, *J. Comput. Phys.*, 67, 124-144.
3. D.C. ARNEY and J.E. FLAHERTY (1989). An Adaptive Local Mesh Refinement Method for Time-Dependent Partial Differential Equations, *Appl. Numer. Math.*, 5, 257-274.

4. D.C. ARNEY and J.E. FLAHERTY (1990). An Adaptive Mesh-Moving and Local Refinement Method for Time-Dependent Partial Differential Equations, *ACM Trans. on Math. Softw.*, 16, 48-71.
5. M.J. BERGER and P. COLELLA (1989). Local Adaptive Mesh Refinement for Shock Hydrodynamics, *J. Comput. Phys.*, 82, 64-84.
6. M.J. BERGER and J. OLIGER (1984). Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, *J. Comput. Phys.*, 53, 484-512.
7. M. BIETERMAN and I. BABUŠKA (1982). The Finite Element Method for Parabolic Equations, *Numer. Math.*, 40, 373-406.
8. F.A. BORNEMANN (1990). An Adaptive Multilevel Approach to Parabolic Equations I: General Theory and 1D-implementation, *IMPACT Comput. Sci. Engrg.*, 2, 279-317.
9. N. CARLSON and K. MILLER (1988). The Gradient Weighted Moving Finite Element Method in Two Dimensions, in *Finite Elements Theory and Application*, 152-164, ed. D.L. DWOYER, M.Y. HUSSAINI AND R.G. VOIGHT, Springer Verlag.
10. J.M. COYLE, J.E. FLAHERTY, and R. LUDWIG (1986). On the Stability of Mesh Equidistribution Strategies for Time-Dependent Partial Differential Equations, *J. Comput. Phys.*, 62, 26-39.
11. E.A. DORFI and L. O'DRURY (1987). Simple Adaptive Grids for 1-D Initial Value Problems, *J. Comput. Phys.*, 69, 175-195.
12. J.E. FLAHERTY, P.K. MOORE, and C. OZTURAN (1989). Adaptive Overlapping Grid Methods for Parabolic Systems, in *Adaptive Methods for Partial Differential Equations*, 176-193, ed. J.E. FLAHERTY, P.J. PASLOW, M.S. SHEPHARD, J.D. VASILAKIS, SIAM Publications, Philadelphia.
13. J.E. FLAHERTY, P.J. PASLOW, M.S. SHEPHARD, and J.D. VASILAKIS, EDITORS. (1989). *Adaptive Methods for Partial Differential Equations*, SIAM Publications, Philadelphia.
14. W.D. GROPP (1987). Local Uniform Mesh Refinement with Moving Grids, *SIAM J. Sci. Statist. Comput.*, 8, 292-304.
15. W.D. GROPP and D.E. KEYES (1992). Domain Decomposition with Local Mesh Refinement, *SIAM J. Sci. Comput.*, 13, 967-993.
16. D.F. HAWKEN, J.J. GOTTLIEB, and J.S. HANSEN (1991). Review of Some Adaptive Node-Movement Techniques in Finite-Element and Finite-Difference Solutions of Partial Differential Equations, *J. Comput. Phys.*, 95, 254-302.
17. P.J. VAN DER HOUWEN and B.P. SOMMEIJER (1980). On The Internal Stability of Explicit m-Stage Runge-Kutta Methods for Large m-Values, *Z. Angew. Math. Mech.*, 60, 479-485.
18. A.K. KAPILA (1983). *Asymptotic Treatment of Chemically Reacting Systems*, Pitman Advanced Publ. Company.
19. J. LANG and A. WALTER (1992). *A Finite Element Method Adaptive in Space and Time for Nonlinear-Reaction-Diffusion-Systems,* Preprint SC 92-5, Konrad Zuse Zentrum für Informationstechnik Berlin.
20. J.M.L. MAUBACH (1991). *Iterative Methods for Nonlinear Partial Differential Equations,* PhD. Thesis, Catholic University of Nijmegen, Netherlands.
21. K. MILLER (1981). Moving Finite Elements II, *SIAM J. Numer. Anal.*, 18,

1033-1057.

22. K. MILLER and R.N. MILLER (1981). Moving Finite Elements I, *SIAM J. Numer. Anal.*, 18, 1019-1032.
23. L.R. PETZOLD (1987). Observations on an Adaptive Moving Grid Method for One-Dimensional Systems of Partial Differential Equations, *Appl. Numer. Math.*, 3, 347-360.
24. K. REKTORYS (1992). *The Method of Discretization in Time, Series Mathematics and Its Applications 4*, Reidel, Dordrecht.
25. M.D. SMOOKE and M.L. KOSZYKOWSKI (1986). Fully Adaptive Solutions of One Dimensional Mixed Initial-Boundary Value Problems with Applications to Unstable Problems in Combustion, *SIAM J. Sci. Statist. Comput.*, 7, 301-321.
26. R.A. TROMPERT and J.G. VERWER (1991). A Static-Regridding Method for Two Dimensional Parabolic Partial Differential Equations, *Appl. Numer. Math.*, 8, 65-90.
27. A.J. WATHEN (1992). Optimal Moving Grids for Time-Dependent Partial Differential Equations, *J. Comput. Phys.*, 101, 51-54.
28. P.A. ZEGELING (1992). *Moving-Grid Methods for Time-Dependent Partial Differential Equations,* PhD. Thesis, University of Amsterdam, Netherlands.
29. P.A. ZEGELING and J.G. BLOM (1992). A Note on the Grid Movement Induced by MFE, *Int. J. for Numer. Meth. in Eng.*, 35, 623-636.

Chapter 2

A Static-Regridding Method for Two-Dimensional Parabolic Partial Differential Equations

R.A. Trompert and J.G. Verwer
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

The subject of this paper belongs to the field of numerical solution of time-dependent partial differential equations. Attention is focussed on parabolic problems in two space dimensions the solutions of which possess sharp moving transitions in space and time, such as steep moving fronts and emerging and disappearing layers. For such problems, a space grid held fixed throughout the entire calculation can be computationally inefficient, since, to afford an accurate approximation, such a grid would easily have to contain a very large number of nodes. We consider a so-called static-regridding method that adapts the space grid using nested, local, uniform grids. The notion of local uniform mesh refinement is an example of 'domain decomposition', the general idea of which is to decompose the original physical or computational domain into smaller subdomains and to solve the original problem on these subdomains. Hence, our computational subdomains are nested, local, uniform, space grids with nonphysical boundaries which are generated up to a level of refinement good enough to resolve the anticipated fine scale structures. This way a fine grid covering the entire physical domain can be avoided. We discuss several aspects occurring in a static-regridding algorithm like ours, such as the data structure, the regridding strategy and the determination of initial and boundary conditions at coarse-fine grid interfaces. We also present two numerical examples to demonstrate the performance of the method. The first example is hypothetical and is used to illustrate the convergence behavior of the method. The second example originates from practice and describes a model combustion process.

published in: Appl. Numer. Math. **8** (1991), 65-90

2.1. Introduction

Many evolution problems involving linear or nonlinear partial differential equations (PDEs) have solutions with sharp moving transitions such as steep wave fronts and emerging or disappearing layers. In such situations, a grid held fixed throughout the entire calculation can be computationally inefficient, since, to afford an accurate approximation, such a grid would easily have to contain an unacceptably large number of nodes. Adaptive grid methods strive to resolve these sharp transitions to acceptable degrees of accuracy while avoiding the use of excessive numbers of grid points. Such methods use, in some way or another, nonuniform or local, uniform grids and, as time proceeds, automatically concentrate the grid in spatial regions of high activity. It thus is attempted to keep the number of nodes at an acceptable level.

For time-dependent problems one may distinguish two main categories of methods, viz., dynamic-regridding and static-regridding methods. In the first category the grid moves continuously in the space-time domain, like in classical Lagrangian methods, while usually the discretization of the PDE and the grid selection are intrinsically coupled. A well-known example is provided by the moving-finite-element and related methods (see, e.g., the proceedings [8]), In the second category the grid is moved only at discrete time levels and no intrinsic coupling exists between the discretization of the PDE and the grid selection. This category comprises a large number of different methods and the method developed in this paper is of the static-regridding type.

Our method is closely related to the methods of *Berger* and *Oliger* [4], *Gropp* [9-11] and *Arney* and *Flaherty* [2] and thus an important characteristic of it is local uniform mesh refinement. The notion of local uniform mesh refinement is an example of 'domain decomposition', the general idea of which is to decompose the original physical or computational domain into smaller subdomains and to solve the original problem on these subdomains. The idea of our static-regridding approach can be briefly described as follows. Given a coarse base space grid and a variable, base temporal step size, a 'decomposition' into local, uniform subgrids is performed recursively. Hence, our computational subdomains are nested, local, uniform, space grids with nonphysical boundaries which are generated up to a level of refinement good enough to resolve the anticipated fine scale structures. The step sizes in time and space during this refinement process are chosen automatically by comparing estimates or indicators of local temporal and spatial errors to prescribed tolerances, while the refinement is carried through until all tolerances are met. The process is then continued to the next base space/time mesh, while all fine grid results computed at forward time levels are kept in storage as these are needed for step continuation.

An attractive feature of moving the points only at discrete time levels is the possibility of dividing the whole solution process into the following computational procedures: spatial discretization, temporal integration, error estimation, regridding and interpolation. Depending on the application, these individual procedures may range from simple or straightforward to very sophisticated. This flexibility is attractive since it makes it possible to treat different types of PDE problems with almost one and the same code, assuming hereby that the grid structure and the associated data structure remain unchanged. In this connection we wish to note that a major part of the development and coding of any static-regridding method, including ours, lies in the grid and data structure. The choice of data structure is important for keeping the unavoidable overhead at an acceptable level, because at each time step grids may be created or removed while also communication between grids of adjacent levels of refinement frequently takes place.

When contrasted with the dynamic approach, an inherent drawback of static regridding is that during the time stepping temporal variations are not minimized because grid points do not move. More specifically, when a steep front passes a fixed grid point, smaller integration steps are required to maintain accuracy than when the grid point travels along with the front in the proper direction. In this connection, interpolation at internal grid interfaces should also be used judiciously for generating sharp solution profiles. Dynamic regridding methods using a fixed number of moving points obviously do not require interpolation at internal interfaces which may be considered as an advantage. On the other hand, a well-known major disadvantage of dynamic regridding is that methods of this type often have difficulty in controlling grid skewness and grid tangling. This disadvantage is usually less in hybrid methods where in some way or another static- and dynamic-regridding concepts are combined. Examples of such methods can be found in, e.g., *Arney* et al [2], *Gropp* [10] and, for 1D problems, in *Petzold* [16] and *Verwer*, *Blom* and *Sanz–Serna* [21]. Finally, an attractive feature of local mesh refinement is that it enables prescribed tolerances to be satisfied by using finer-and-finer grids in regions where greater resolution is needed.

In this paper we concentrate on (systems of) parabolic equations of the reaction-diffusion type,

$$\begin{aligned} u_t &= \mathcal{L}(x,y,t,u) := d(u_{xx} + u_{yy}) + f(x,y,t,u), \qquad (1.1) \\ u &= u(x,y,t), \quad (x,y,t) \in \Omega \times \{t > 0\}, \end{aligned}$$

subjected to appropriate initial and boundary conditions. For simplicity and convenience of presentation, our procedures are described and implemented on rectangular Ω, but they would also apply if Ω is a union of rectangles or can be transformed this way. Throughout the development of the method, the actual form of the operator $\mathcal{L}$ and its boundary conditions play no essential role. In fact, since for spatial discretization the use of standard finite differences is supposed, a much wider class of differential operators $\mathcal{L}$ is allowed.

The contents of the paper is as follows. We start with an outline of our local uniform mesh refinement algorithm in Section 2. The various components of the solution algorithm are discussed in greater detail in the following sections. This discussion includes the actual refinement strategy (Section 3), the data structure (Section 4) and the grid interface conditions (Section 5). For the temporal integration we advocate one-step Runge-Kutta methods, since linear multistep methods like BDF are less appropriate due to the start up problem. Section 6 is devoted to a particular Runge-Kutta method, viz., the explicit Runge-Kutta-Chebyshev (RKC) method of *van der Houwen* and *Sommeijer* [13]. The temporal integrator we have used in the present investigation is based on this method, but it should be stressed that other choices of Runge-Kutta methods are possible too. The error indicators that govern the selection of the step sizes in time and space are discussed in Section 7. In Section 8 we present two examples of reaction-diffusion problems (1.1) that were solved with our static-regridding method using the explicit RKC scheme for time integration. One of these two example problems is nonlinear and originates from combustion theory. Our future plans are summarized in Section 9.

2.2. Outline of the Algorithm

In this section we present a rough outline of our recursive static-regridding algorithm, so as to facilitate the presentation and discussion of algorithmic details in later sections. Our algorithm is based on the principle of local uniform mesh refinement (LUMR). LUMR may be contrasted with pointwise refinement which leads to truly nonuniform grids. As already noted in the introduction, LUMR is an example of 'domain decomposition'. When considered this way, our domain is a base space/time grid determined by a base space grid consisting of rectangular quadrilateral cells with sides Δx, Δy for $T \le t \le T + \Delta t$. The temporal step size Δt is called the base temporal step size and the base space grid covers the physical domain. It is assumed in this paper that this domain is rectangular, but without essential changes the domain is allowed to be made up of a union of rectangles (polygonal boundary parallel to the co-ordinate axes). The 'domain decomposition' or regridding on local, uniform grids now takes place entirely within this base space/time grid defined on the time interval $T \le t \le T + \Delta t$. Starting at the physical initial time, the complete algorithm is then repeatedly applied until the desired final physical time is reached. We will outline all computations required for advancing the solution at the base grid at time T to the next base grid at time $T + \Delta t$. This outline is similar as in *Gropp* [11]. The base grid parameters Δx, Δy are supposed to be prescribed. The base temporal step size Δt is supposed to be a trial value. For clarity we discuss first a single level of refinement where the coarse grid coincides with the base grid.

The following steps are followed to advance the solution from time T to the next base time level $T + \Delta t$:

(1) Integrate on the coarse grid using one coarse time step of size Δt. Adaptation in space is always preceded by adaptation in time, that is, the time step is first subjected to a temporal local error test and eventually the integration is redone with a smaller Δt until acceptance takes place. Call the accepted values of u on the coarse grid at time $T + \Delta t$ the new coarse u-values and those at time T the old coarse u-values. Both sets of values are saved.

(2) Integration is followed by regridding. Using the new coarse u-values, decide where the fine grid will be for $T \le t \le T + \Delta t$. This is done by invoking a spatial local error indicator and a clustering and buffering algorithm to distribute all untolerable cells over the fine grid. The fine grid may consist of different disjunct fine subgrids. Overlapping fine subgrids are not allowed in our method and fine subgrids need not be a rectangle. The actual refinement is cellular and carried out by bisecting all sides of untolerable cells. At this point the non-refined part of the coarse grid is complete and not further processed within the current coarse time step.

(3) Regridding is followed by interpolation. Return to time level T and determine initial values for the fine grid. If a cell was refined in the previous coarse time step, then we use the available fine grid u-values and interpolation is not needed. If a cell was not refined before, then we interpolate old coarse u-values. For $T \leq t \leq T + \Delta t$ we need to specify boundary values at grid interfaces where fine grid cells abut on coarse cells. Using old and new coarse u-values, at these grid interfaces numerical Dirichlet boundary conditions are prescribed via interpolation. If an interface coincides with the physical boundary, then physical boundary conditions are used.

(4) Next the fine grid is integrated over the interval $T \leq t \leq T + \Delta t$ while fine grid u-values are subjected to the temporal local error test. This adaptation in time may result in a smaller temporal step size than Δt. If $T + \Delta t$ is reached, then the new fine grid u-values are injected in the coarse grid points, i.e., the value of u at $T + \Delta t$ on the coarse grid is taken to be the value of u just computed on the fine grid. Further, all fine grid u-values values at $T + \Delta t$ are saved for use in the next coarse time step. The solution at time $T + \Delta t$ is now complete.

Multiple levels are handled in a natural, recursive fashion. After each accepted time step of (4), taken on a grid of refinement level l, say, a regridding may take place resulting in a grid of refinement level l+1. In fact, the computational steps follow precisely points (2) - (4) above. Note that all fine grid results at forward time levels are kept in storage and that for step continuation the most accurate results are used that are available.

An illustration of the recursive local refinement, in 1D for simplicity of presentation, can be found in Figure 2.1. Including the base grid there are three levels of spatial refinement. In this figure the temporal step size is halved when going to a finer grid. The numbers next to the individual grids indicate the order in which the solution was computed. Herewith we do not distinguish between disjunct subgrids at the same level of refinement. Only the order of the time integrations is indicated. If in this example the base space grid is given refinement level 1, then the order of the corresponding occurring refinement levels is 1, 2, 3, 3, 2, 3, 3. The next occurring refinement level is 1, assuming that the next time step is taken on the next base grid. For clarity, a time history diagram of the occurring refinement levels is given in Figure 2.2. The location of the fine grids is not made visible in this time history diagram and thus it may also correspond with a base time step of a 2D computation. With Figures like 2.1 and 2.2 one may now easily conceive other possible order of refinements. Finally, Figure 2.3 shows a two-dimensional example of a base grid with three fine subgrids.

In conclusion, the principle on which our static-regridding method is based is recursive LUMR and cellular refinement. The building blocks of our algorithm are nested, local, uniform, finer-and-finer grids which are adaptively defined by invoking indicators for local spatial and temporal errors. These local uniform fine grids need not be rectangles and their location in the base space/time domain is automatically governed by the error indicators. Initial and boundary conditions for a refined subgrid are taken from the parent coarse grid or from the given initial function and physical boundary conditions. The recursive refinement takes place repeatedly per coarse time step. Solutions computed on current fine grids are kept in memory for use in the next coarse time step.

Inherent in the approach we have adopted is that grid information is not passed to the next coarse time step. This necessarily is a bit wasteful in situations where the sharp transitions move very slowly, e.g., when approaching steady state. On the other hand, the computational effort for the coarser grids normally shall not be large. Further, uniform subgrids allow an efficient use of vector based algorithms and finite-difference or finite-element expressions on uniform grids are more accurate and cheaper to process than on nonuniform grids. In this respect the current LUMR approach should be contrasted with pointwise refinement where arbitrary levels of refinement around any point are allowed. This pointwise refinement leads to truly nonuniform grids on which usually less points are needed than on an LUMR grid. However, an inherent drawback of a truly nonuniform grid is a more complex and expensive data structure. We refer to *Gropp* [9-11], *Berger* [3] and *Ewing* [7] for some further discussions on various advantages and/or disadvantages when comparing the two refinement techniques. We also note that the LUMR methods of *Berger* and *Oliger* [3,4] (see also *Arney* and *Flaherty* [2]) are based on noncellular refinement and truly rectangular subgrids which may overlap and rotate to align with an evolving dynamic structure.

2.3. Refinement strategy and grid structure

When a new refinement level is to be created, the new fine grid has to meet two demands. First, the new fine grid should be as efficient as possible, that is, no cells are unnecessarily refined. This implies that the new fine grid is allowed to have an irregular shape and also may consist of different disjunct subgrids. For clarity, it is noted once more that in the discussion we do not distinguish between subgrids belonging to the same level of refinement. In other words, when we write 'new fine grid', we mean the complete grid associated to the next higher refinement level and this grid may consist of different disjunct subgrids. The second demand is that the accuracy at interior nodes of a new fine grid should not be diminished by low accuracy nodes on its boundary. Therefore, the boundary of the new fine grid must either coincide with a physical boundary or lie in part of the physical domain where the accuracy is sufficiently high, relative to the error measurement used. This second demand is most important because when at a certain level a cell is not further refined, we never return to this cell within the current base time step. In this section we describe a local refinement strategy that conforms to both demands.

The refinement strategy decides where a new fine grid will be placed (cf. point (2) of Section 2). The refinement is governed by a so-called local spatial error indicator, ests, and a corresponding tolerance value, tols, that has to be specified. Ideally, ests estimates the genuine local spatial truncation error that has been committed on the grid currently in use. We discuss the actual choice of the indicator in Section 7. For the discussion of the present section it suffices to suppose that we are given values

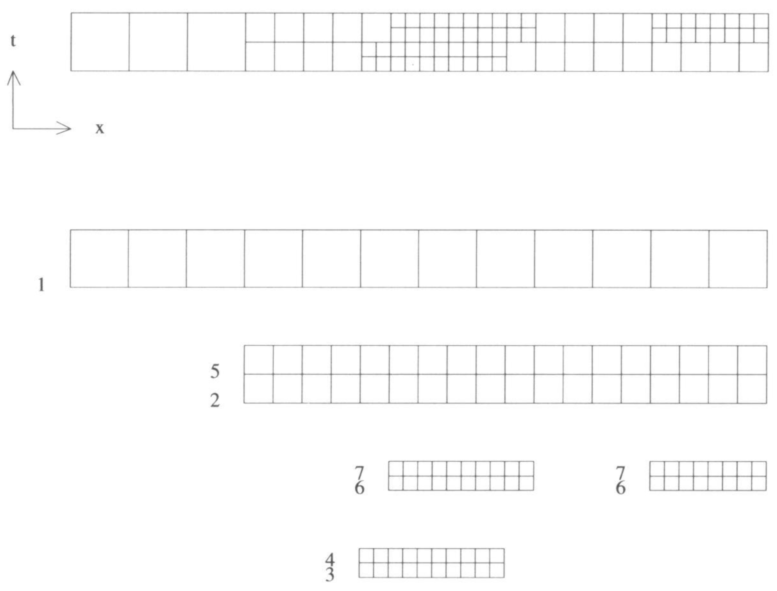


FIGURE 2.1. Typical set of LUMR grids in one space dimension for one base time step. The final composite grid is shown at the top of the figure. Note that here the step sizes in time are halved. In actual application the new step size is determined by the local time error indicator.

of ests at the current forward time level at all nodal points of the grid currently in use.

Let l be the level index of this grid. Let estsm be the maximum of all ests values. If

$$\text{estsm} > \text{tols}, \tag{3.1}$$

then it is decided to create a new fine grid of level $l+1$. For this purpose, a second spatial tolerance value is introduced. It is this second tolerance, denoted by tolspc and derived from tols, that is used to decide which particular level-l cells need to be refined. The second tolerance tolspc is defined as follows. Let p be the order of consistency of the spatial discretization and of the local error indicator. In our case $p = 2$ since we here work with standard finite differences. Since the mesh width of level $l+1$ is half the mesh width of level l, etc., we may invoke the asymptotic order relation

$$\text{estsm}(k) = 2^{-p(k-l)}\text{estsm}(l), \quad k \geq l+1, \tag{3.2}$$

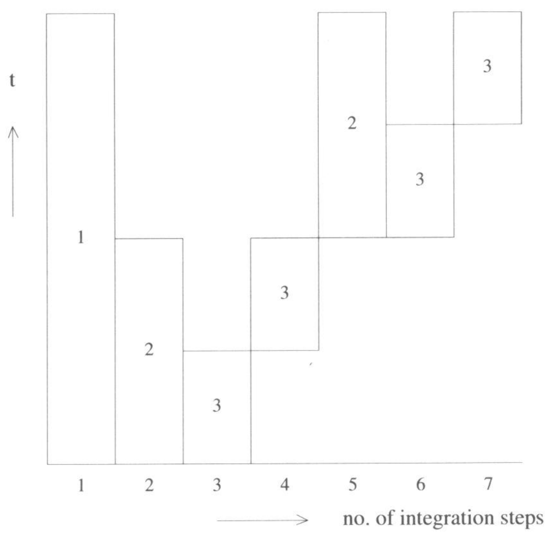


FIGURE 2.2. Time history diagram of refinement levels occurring in Figure 2.1. Each bar corresponds with an integration step. The levels are indicated within the bars. The lower and upper line of a bar correspond with the old and new time value of the integration step. The actual location of the refined grids is not shown.

where estsm(l) represents estsm of the level-l grid, etc., and k is a grid level index as yet unknown. We here anticipate that on top of level l, another $k-l$ refinement levels will be needed to satisfy the condition

$$\text{estsm}(k) \leq \text{tols}. \tag{3.3}$$

From this inequality the unknown integer k is now computed, that is,

$$k = l + 1 + \text{entier}[(\log(\text{estsm}(l)) - \log(\text{tols}))/(p\log(2))]. \tag{3.4}$$

The idea is now to impose, at all nodal values of level l, the refinement condition

$$\text{ests} > \text{tolspc} := \text{estsm}(k), \tag{3.5}$$

hereby introducing the second spatial tolerance value. However, because the calculation of a spatial error indicator comes down to the calculation of higher derivatives by means of finite differences (numerical differencing), it is conceivable that these

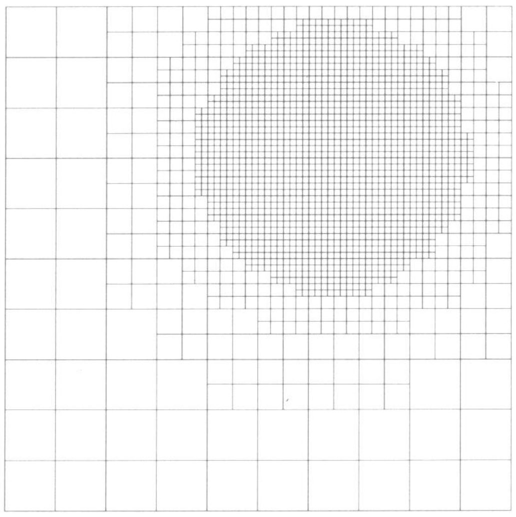


FIGURE 2.3. Example of 2D grid structure. It is emphasized that fine grids are not patched into coarse ones, but overlays them. The time dependency is not shown here.

are underestimated in regions where the solution is steep. Therefore, by way of safety, we suppose that one extra level would be preferable, which means that in (3.5) a safety factor of 2^{-p} must be built in. Thus we finally arrive at the refinement condition

$$\text{ests} > \text{tolspc} := 2^{-p}\text{estsm}(k). \tag{3.6}$$

This condition is used to decide which level-l cells need refinement. The rationale behind this second tolerance value is that we wish to refine all level-l cells, except those for which ests does not exceed the expected maximum of the error indicator values at the anticipated highest refinement level k. This local refinement strategy takes into account the method strategy that when at a certain level a cell is not refined, we never return to this cell within the current base time step. Another natural justification is that, when using local mesh refinement, this strategy attempts to have the maximum norm of the spatial error over the complete physical domain equal to or smaller than the maximum norm of the spatial error over local grids.

The actual cell refinement goes as follows. Any level-l node satisfying (3.6) is flagged together with its eight neighboring nodes. Next, to create an extra buffer, all sides of cells with at least one flagged corner node are bisected. This means that a buffer zone of two coarse or four fine mesh widths is used around any untolerable node. Hence, the minimal number of nodes in a column and row of any subgrid is nine. Herewith it is tacitly assumed that the minimal number of internal points in a row and column of the coarsest base grid is three. Near boundaries, physical and internal ones, the buffering of course slightly differs. Finally, a cluster algorithm groups all untolerable cells together to form the newly defined level-l+1 grid. It is noted that subgrids in the new fine grid do not overlap and that we do not connect subgrids which are lying close together since this leads to substantial bookkeeping. For the same reason, the local refinement does not distinguish between co-ordinate directions. This necessarily leads to some waste of points if a high gradient region aligns with a co-ordinate direction.

The use of tolspc in combination with the buffer zone is rather conservative. However, to our experience, it nicely conforms to the second demand stated above that accuracy at interior nodes of finer-and-finer grids should not be diminished by a too low accuracy at nodes on a previously selected interior boundary.

2.4. Data structure and memory use.

In this section we briefly discuss the data structure we have implemented. Our data structure has close similarities with that of sparse matrix storage schemes, in particular concerning the storage of a sparse matrix as a collection of sparse vectors. The interested reader is referred to *Duff* et al. [6], Ch. 2, for the various technical details that are involved in the use of this type of storage schemes.

The data we keep for each node are stored per level of refinement in a row sequential order. In particular, rows in subgrids are taken together and a 'sparse' vector in our storage scheme contains all data belonging to all nodes of a row at a certain level. The data we store for each level of refinement are as follows:

- The number of rows.
- For each row
    - A row index corresponding with its y-co-ordinate.
    - A pointer to the memory location of the data belonging to its first node.
- For each node
    - A column index corresponding with its x-co-ordinate.
    - Two pointers to memory locations where data belonging to the nodes directly above and below in the same grid is stored.
    - An integer indicating the position of a node in the domain, i.e., whether it lies on the physical boundary, on an internal boundary, or in the interior of the grid.
    - The solution and its time derivative at the beginning and end of a time step.

The row and column indices are, amongst others, used to find coinciding nodes on different refinement levels. This is needed for interpolation and injection. The row and column indices correspond to the x- and y-co-ordinates in the following way. Let i be the column index and j the row index of a node at refinement level l. Then,

$$x = x_0 + i\Delta x \, 2^{-(l-1)}, \qquad y = y_0 + j\Delta y \, 2^{-(l-1)}, \tag{4.1}$$

where Δx, Δy are the coarsest level mesh widths and (x_0, y_0) is the co-ordinate of the left lower corner of the physical, rectangular domain. It follows that a node on level l with column index i and row index j coincides with a node on level l+1 with column index $2i$ and row index $2j$.

We will now outline how the actual storage is organized. The arrays used are divided in a number of blocks of the same size. The number of blocks equals the maximum of the number of refinement levels that is expected to be required during actual runtime. Each of the blocks is assigned a refinement level and all data for this refinement level are stored in this block. Hence the size of the blocks should be large enough to store all generated data of any refinement level that can occur during runtime. Herewith we implicitly determine the maximum number of levels and the maximum number of rows and nodes per level. The drawback of using memory blocks of the same size is that memory is wasted, as it is likely that the actual amount of memory needed differs per level. A clear advantage is computational simplicity, because there is no need for garbage collection. Because the size of the blocks is fixed, their initial subscript values are known which makes the data retrievement simple and fast. Nevertheless, the implementation of our algorithm is such that a garbage collector can be easily implemented. In this connection it is noted that with minor modifications our storage scheme can be extended to 3D. Needless to say that in 3D the memory waste could be prohibitive so that garbage collection becomes a necessity. All experiments reported in this paper have been carried out without garbage collection, indicating that in 2D this is not much of a problem.

Next we give an estimate of the amount of memory needed in bytes. Work arrays needed by the regridding algorithm and the time integrator are not included in this estimate. The work arrays for the regridding algorithm are integer arrays and their total length is negligible compared to the estimate given below. Further, for the time integrator we use one and the same set of work arrays for all levels. The total length of these arrays is of course determined by the actual integrator in use (see Section 6).

The array in which we store the solution and its time derivative at the begin and end of an integration step is a real floating point array. The number of bytes per node this array uses is 32 times the number of PDEs (NPDE), assuming precision arithmetic involving 64 bits. The four arrays holding the column indices, the pointers to nodes above and below, and the position indicators, are 4-bytes integer arrays and together they take 16 bytes per node. The two arrays containing the row indices and the memory addresses of the data belonging to the first node of all rows are also integer arrays. These row indices and starting addresses require together 8 bytes of memory per row. Because the minimum number of nodes on a row is 7, it follows that these two arrays take at most 8/7 bytes per node. We thus arrive at a final estimate in bytes given by

$$\text{maxlev} * \text{nptspl} * (32 * \text{NPDE} + 16 + 8/7), \tag{4.2}$$

where maxlev is the maximum number of levels and nptspl is the maximum number of nodes per level.

2.5. Interface conditions

When a new fine grid is created, initial and boundary values have to be defined. Concerning the initial values, three cases are distinguished. If the time level at which initial values must be specified coincides with the physical initial time, then of course the prescribed initial function is used at any occurring node. If the time level does not coincide with the physical initial time, then two possible cases remain. A node coincides with a node at the same refinement level from a previous time step. In this case we adopt the available solution at this node as initial value. In the other case we must interpolate. The initial value is then obtained from interpolation on the nearest coarse grid. Obviously, the interpolation error should not diminish the accuracy. We use fourth-order Lagrangian interpolation and note that this way second-order accuracy is maintained in calculating the first and second spatial derivatives with the second-order difference scheme. Note that we use an explicit integration scheme which starts with a spatial differential operator calculation at the initial time level (stage (6.2b)). It is then prohibited to use straightforward linear interpolation, since this would yield zero second-derivative values at the newly created nodes. When using implicit methods, this difficulty can be avoided by selecting the method such that no evaluation at the initial time is used.

Concerning boundary values, again three possible cases are distinguished. A boundary node of the new fine grid is located on the physical boundary. In this case the physical boundary condition is imposed. If a new boundary node belongs to the interior of the physical domain, still two different cases are possible. First, the node coincides with a node of the nearest coarse grid. Then we define Dirichlet boundary values by interpolating at the beginning and end of the new time interval to be covered. This time interval always coincides with a previous coarse grid time step. Fourth-order Hermite interpolation is applied, using the available solution and first time derivative from the coarse grid. Second, the node is new and hence does not coincide with a node of the nearest coarse grid. In this case a solution value is already available at the initial point of the new time interval by the above Lagrangian interpolation procedure. To be able to apply again Hermite interpolation as outlined above, the solution and time derivative at the end point of the time interval are also generated by Lagrangian interpolation on the coarse grid.

Because we use second order in space and time in the discretization of the PDE, the fourth-order interpolation procedures should be sufficiently accurate, provided of course the local refinement has been carried through far enough at the time of the interpolations. Note that the prescription of interior Dirichlet boundary values is natural, since we solve initial boundary value problems involving second-order spatial differential operators.

2.6. Runge-Kutta-Chebyshev method

We will now describe the temporal integrator used for the experiments reported in this paper. For this purpose we introduce the system of ordinary differential equations (ODEs)

$$\frac{dU(t)}{dt} = F(t, U(t)), \tag{6.1}$$

assuming that this system originates from spatial discretization of the PDE problem (1.1) on the coarse base grid or on a refined local grid (method of lines). It is also assumed that boundary conditions, physical and artificial, have been incorporated into the continuous time, semi-discrete form (6.1). At this stage of development there is no need being more specific about (6.1).

The integration method is the explicit Runge-Kutta-Chebyshev (RKC) method of *van der Houwen* and *Sommeijer* [13]. This method has been designed for the numerical integration of large stiff systems of ODEs which originate from spatial discretization of multi-space dimensional parabolic PDEs such as (1.1). We will present only a brief outline here. More details can be found in [13] and *Verwer, Hundsdorfer* and *Sommeijer* [20]. The latter paper is devoted to a convergence analysis of the RKC method.

Let U_n denote an approximation to $U(t)$ at time $t = t_n$. Let Δt denote the step size in time. The next approximation U_{n+1} at time $t_{n+1} = t_n + \Delta t$ is then given by the s-stage integration method

$$\begin{aligned}
Y_0 &= U_n, && \text{(6.2a)}\\
Y_1 &= Y_0 + p_1 \Delta t F_0, && \text{(6.2b)}\\
Y_j &= m_j Y_{j-1} + n_j Y_{j-2} + (1 - m_j - n_j) Y_0 + p_j \Delta t F_{j-1} + q_j \Delta t F_0, && \text{(6.2c)}\\
&\quad 2 \le j \le s, &&\\
U_{n+1} &= Y_s, && \text{(6.2d)}
\end{aligned}$$

where $F_j = F(t_n + c_j \Delta t, Y_j)$ and the value Y_j represents an intermediate, auxiliary approximation to $U(t)$ at the time point $t = t_n + c_j \Delta t$. Thus, the RKC method is to be interpreted as a one-step, s-stage Runge-Kutta method.

The available method parameters are used for obtaining a very large real interval of stability. This is achieved by identifying the recursive formula (6.2c) with a stable three-term Chebyshev recursion, thus explaining the specific form of the integration method. The length of the real stability interval that is obtained this way is proportional to s^2, where the proportionality constant depends on the order of consistency and on a damping property imposed on the common stability function. This stability function is a shifted Chebyshev polynomial.

The notion of stability meant here is the common linear stability based upon the linear system

$$\frac{dU(t)}{dt} = F(t,U(t)) := MU(t) + f(t), \quad M \text{ symmetric.} \tag{6.3}$$

Specifically, we have stability, in the step-by-step sense, if Δt and s are such that the inequality

$$\Delta t \; \sigma(M) \le \beta(s), \tag{6.4}$$

is satisfied, where $\sigma(M)$ represents the spectral radius of M and $\beta(s)$ is the real stability boundary of the method. We have worked with a second-order method (formulas (2.19) - (2.21) from [20]) of which the real stability boundary is given by

$$\beta(s) \approx \frac{2}{3}s^2. \tag{6.5}$$

If the system (6.1) is nonlinear, then criterion (6.4) is imposed in the common heuristic way. Specifically, M is then understood to represent the Jacobian matrix of the vector function F taken at an appropriate point. Experience has revealed that the linear theory is most reliable if F stems from a nonlinear parabolic problem and the Jacobian matrix is symmetric. The method is recommended only if the Jacobian is symmetric or 'nearly symmetric'. This excludes, e.g., convection-diffusion problems with dominating convection.

The RKC method is applied with variable step size governed by a local error indicator (cf. Section 7) and with a variable number of stages s, such that always the linear stability inequality (6.4) is satisfied with s minimal. The variable s-strategy leans upon two properties. First, the error indicator is independent of s. Second, thanks to an internal stability property associated with the three-step Chebyshev recursion, there is no practical limit on s. This in fact implies that the method can be applied as if it is unconditionally stable, simply by adjusting s at each integration step to satisfy (6.4) for given step size and given spectral radius. The computation of (a safe upper bound) of the spectral radius normally renders no problem for operators like (1.1).

When compared with the implicit approach, explicitness has an inherent advantage for static regridding, since the costs of the numerical algebra involved in the application of implicit methods is much larger than in standard (single-grid) method-of-lines applications. Recall that at any time step a regridding may take place at different levels of refinement, thus introducing one or more new Jacobians of different order at any time step when using an implicit method. This degrades the efficiency of stiff ODE solvers, since these often benefit from integrating with old Jacobians over many time steps. In spite of this, there are of course many problems and situations where for stability reasons alone implicit time stepping becomes a necessity. Therefore, as a continuation of the research reported here, in the near future we will investigate the application of implicit Runge-Kutta methods for static regridding. An important aspect hereby is the choice of appropriate implicit equation solvers which can efficiently deal with various types of 2D systems (see, e.g., *Hindmarsh* and *Nørsett* [12]).

Concerning stability, the explicit RKC method should be positioned between classical explicit methods yielding a severe time step restriction and unconditionally stable implicit ones. For problems with symmetric Jacobians, the RKC method is still attractive in cases of substantial stiffness due to the quadratic dependence of the real stability boundary on s. Furthermore, the method is simple to implement and the explicitness offers natural prospects for vector-based implementations. Also the memory demand is low. We have used a variable step size FORTRAN code due to *Sommeijer* [19] that needs only 6 arrays of storage.

It is noted that our version of this code slightly differs from Sommeijer's original one. This concerns merely the local error indicator. As we will outline in the next section, the third-derivative estimator of the original code, based on the genuine local truncation error, does not function well in our adaptive grid application and has therefore been replaced by a more simple error indicator. However, the actual strategy associated with varying the step size, threshold factors and the like, has not been changed. It would lead us too far here to discuss this strategy in detail and it suffices to remark that, apart from the error indicator itself, the variable step size strategy is conceptually similar to strategies in existing ODE codes.

2.7. Error indicators

In static-regridding methods three different kinds of local errors show up, viz., spatial discretization errors, time integration errors and interpolation errors. The asymptotic behavior of the local space and time errors for decreasing spatial and temporal grid sizes is well understood for single-grid applications. This is also true for the propagation of resulting global errors, at least for interesting model situations (see e.g. [17, 20]). Needless to say that the static regridding complicates the error analysis considerably. Such an analysis should provide insight into how these three different local errors interfere with each other and propagate or accumulate under regridding. Subsequently, this insight then should assist us in the choice of mathematically correct, practical local error estimators. Without good estimators it is likely that one wastes computational effort due to bad balancing of space and time errors. This in fact is also true for standard, single-grid method-of-lines applications (see *Berzins* [5] who studies the balancing of space and time errors in applications using the BDF method). However, the question of balancing space and time errors is most interesting for a static-regridding method, because for such a method the regridding apparatus is available and it is natural to let the local refinement in space be governed by the genuine space truncation error.

In a sequel to this paper we will present a convergence analysis of the static-regridding method. This analysis is supposed to cover both explicit and implicit

Runge-Kutta methods for the time integration and shall be aimed at practically balancing genuine local space and time errors. Here we confine ourselves to illustrating the convergence of the method numerically (next section) and to using simple heuristic, local error indicators. These error indicators are cheap and function quite satisfactorily from the point of view of ease of use, viz., they automatically invoke refinement in regions with high gradients. On the other hand, they do not guarantee a good balance between spatial and temporal local errors.

Let us first define ests, the spatial error indicator first introduced in Section 3. In this paper ests is based on the 'curvature' expression,

$$(\Delta x)^2\,|u_{xx}| + (\Delta y)^2\,|u_{yy}|, \tag{7.1}$$

which is computed with the three-point finite-difference scheme. To our experience, this 'curvature' expression functions well in measuring the degree of spatial difficulty of the problem. We have also used it successfully in 1D moving grid computations [21]. Note that the genuine space truncation error is also of second order in the mesh widths, since we use the second-order finite-difference scheme for spatial discretization. We thus have proportionality between the tolerance tols and the spatial truncation error.

We next define estt, the temporal local error indicator that determines the step size in time. According to the outline presented in Section 2, estt is computed after every time step at any grid level. Naturally, the choice of estt should be determined by the integration method (cf. the discussion at the end of Section 6). To our experience, the original third-derivative estimator of the RKC method functions quite satisfactorily in standard, fixed grid applications. However, we have encountered difficulties with this estimator in our adaptive grid application. These difficulties are inherent to static regridding and similar to those reported by *Petzold* [15, 16].

The following observations are in order. The regridding implies that components of a solution vector that acts as the initial vector for a following time step may be of three different types. Components may result directly from a preceding integration step on the same grid level, components may result from injection from a higher preceding level, and components may be obtained from interpolation at the next coarser level. While it is supposed that the accuracy is not adversely affected by the interpolation and, trivially so, by the injection, the effect of the regridding thus is that small 'discontinuities' are introduced into the initial vector, which in turn introduce small stiff transient solution components in time. These small transients are damped by the stability of the numerical method. They are, however, seen by common local error estimators like the one implemented in the original RKC code. This estimator computes an approximation to the third solution derivative from solution data that has become available within the current time step. It suffices to recall that this computation is to be interpreted as explicit numerical differencing which holds true for any common local error estimator. The estimators then have a tendency to choose step sizes which are smaller than what is really required to maintain the accuracy, since on nonsmooth data explicit numerical differencing tends to overestimate higher derivatives. When considered on its own, this is not so much of a problem. However, it may also result in unnecessary step rejections which of course should be avoided. For a static-regridding method using different levels of refinement, the sensitivity of the local error estimator for the observed nonsmoothness is even worse, because this nonsmoothness increases for decreasing spatial mesh widths. This implies that irrespective of the strategy used, there will always be a tendency to use too small step sizes on fine grids, which by themselves are already more expensive to process than coarser grids.

For implicit solvers an often used remedy to the problem of nonsmooth error estimates is 'implicit filtering' by which the original, explicit local error estimate is 'smoothed' in an implicit way [15]. This filtering step kills all high-frequent components in the estimated error and leaves, up to $O(\Delta t)$, the low-frequent components of the error unchanged. Because we work here with the explicit RKC scheme, 'implicit filtering' cannot be used. We do not advocate explicit smoothers, since these only work well if the errors to be smoothed do have a regular structure. This is obviously not the case with static regridding, as already mentioned above. To circumvent the difficulty, we therefore base our variable step size on the simple, heuristic local error indicator

$$\text{estt} = (\Delta t)^2 \,|\, u(t+\Delta t) - u(t) \,| = (\Delta t)^3 \,|\, du(t)/dt \,| + O((\Delta t)^4), \qquad (7.2)$$

which is of the same order in step size as the local truncation error estimated in the original code. The rationale behind this first-derivative indicator is that this way we ignore the small, high-frequent error components which we wish to ignore for step size prediction purposes. To our experience, the indicator is successful in this respect. Usually, the imposed tolerance tolt should be chosen smaller than when using the original estimator. This is to be expected, since for solutions with steep temporal gradients, the third derivative shall be larger than the first one. It may also be advisable for keeping ahead of instabilities. In a time stepping process, emerging instabilities result in high-frequent error components which are detected by the local error estimator. The estimator reacts by reducing the step size and consequently restores stability. As indicated, the error indicator (7.2) detects high-frequent error components later than the original estimator. Needless to say that the combination (7.1) - (7.2) is inappropriate for subtle error balancing purposes. However, when using an explicit method like RKC, they form a good compromise because they are cheap and prevent the step size selection from being hindered by the inherent nonsmoothness of the numerical solution.

As already mentioned at the end of the previous section, we have used the existing variable step size code of Sommeijer and have only replaced the existing estimator by (7.2). The step size selection strategy within the code has not been altered. When comparing the estimates with tolt, the maximum norm is used. Finally we mention that the step size prediction is carried out per level. More precisely, when entering a new level, the last predicted step size taken on that level is used. The step sizes are always fitted to hit the endpoint of integration. For example, if at level 2 the predicted step size is smaller than that currently in use at the coarsest level 1, but greater than half this step size, then the level-2 step size is taken to be half of the level-1 step size. During the first base time step the initial step size is the same for any level that may be introduced.

To illustrate the foregoing we conclude this section with a pseudo FORTRAN description of the entire LUMR algorithm. In this description T (level) denotes the end time of the 'level' level. For level = 1 the end time is simply the final physical time. For level > 1 this end time changes dynamically with the introduction and removal of the refined grids and is always smaller than or equal to the forward time value of the current coarse grid step size:

```
        Program LUMR
        level = 1
        T (level) = physical end time
        t (level) = physical initial time
        Δt (level) = initial step size
        call PDEsol
        end LUMR

        subroutine PDEsol
10      if t (level) < T (level) then
20          advance on 'level' level from t (level) to t (level) + Δt (level)
            compute time error indicator estt and predict new Δt (level)
            if estt > tolt then
                decrease Δt (level)
                go to 20
            end if
            compute space error indicators ests and estsm for all nodes at "level"
            if estsm > tols then
                compute tolspc
            end if
            flag all untolerable nodes
            if nodes are flagged then
                generate new 'level' level + 1
                t (level+1) = t (level)
                Δt (level+1) = Δt (level) or previously predicted Δt (level+1)
                T (level+1) = t (level) + Δt (level)
                level = level + 1
                go to 10
            else
                t (level) = t (level) + Δt (level)
                Δt (level) = new Δt (level)
                go to 10
            end if
        end if
```

```
if level > 1 then
    update the solution on 'level-1' with 'level' values at coinciding
        nodes
    level = level - 1
    $t$ (level) = $t$ (level) + $\Delta t$ (level)
    $\Delta t$ (level) = new $\Delta t$ (level)
    go to 10
end if
end PDEsol
```

2.8. Numerical examples

We present two examples of parabolic problems (1.1) that were solved with our static-regridding method using the explicit RKC scheme for time integration. The entire code is written in standard FORTRAN and the numerical experiments have been carried out on a SUN/SPARC station 1.

2.8.1. Problem I

This test problem is hypothetical and due to *Adjerid* and *Flaherty* [1]. The equation is the linear parabolic model equation

$$u_t = u_{xx} + u_{yy} + f(x,y,t), \quad 0 < x,y < 1, \quad t > 0, \tag{8.1}$$

and the initial function at $t = 0$, the Dirichlet boundary conditions for $t > 0$, and the source term f are selected so that the exact solution is

$$u(x,y,t) = \exp(-80((x - r(t))^2 + (y - s(t))^2)), \tag{8.2}$$

where

$$r(t) = \tfrac{1}{4}[2 + \sin(\pi t)], \quad s(t) = \tfrac{1}{4}[2 + \cos(\pi t)]. \tag{8.3}$$

This solution is a cone that is initially centered at $(\frac{1}{2},\frac{3}{4})$ and that symmetrically rotates around the center $(\frac{1}{2},\frac{1}{2})$ of the domain in a clockwise direction. The speed of rotation is constant and one rotation has a period of 2. This problem is not a very difficult one in the sense that the spatial gradients of the solution are not extremely large, that is, the cone is not that steep. However, the problem is suitable to subdue an LUMR regridding algorithm like ours to a convergence test.

The solution is computed five times over the time interval [0,0.25] with an increasing number of levels and decreasing time step. In the first computation only one level is used, in the second two, and so on. The addition of a new level is governed by selecting tols appropriately. Of course, the movement of the refined grids is governed by the regridding algorithm itself. For each computation a constant time step is taken which is the same for all levels (the time step control was switched off). This time step is halved in the next computation when a new level is added. Thus, in view of our regridding strategy based upon using the second tolerance parameter tolspc, we expect the error to decrease with a factor 4, due to the second-order spatial differencing and the second-order consistency of the RKC method. Note that the number of explicit Runge-Kutta stages varies with Δt and the spatial mesh width according to formulas (6.4) - (6.5). We refer to [20] for a convergence test using the fixed grid approach (one single level) where the second order nicely shows up.

Results of the convergence test have been collected in Table 8.1 and Figure 8.1. The figure shows two grid structures used in the level-4 run. Observe that (away from the physical boundary) the grids accurately reflect the symmetry of the rotating cone. This gives confidence that the principles underlying the grid strategy of Section 3 work out very satisfactorily. The size of the region of refinement is still considerable, even at the fourth level. This is due to the fact that the cone is not that steep, as we mentioned earlier.

Table 8.1 contains global errors at specified times given in the maximum norm over the base grid (level 1). The base grid is uniform with a grid distance of 0.1. We observe that the accuracy nicely decreases with the number of levels. Inspection of all output data also revealed that the maximum error was indeed found on the finest grid, as we anticipated in the development of the regridding strategy in Section 3. However, quite interestingly, we also observe a slight reduction in order for increasing number of levels. For instance, at $t = 0.25$ the successive ratios of the errors are approximately 4.8, 3.5, 3.3, 3.2 and thus tend to deviate from 4, although the deviation itself settles down. We believe this order reduction to be inherent in the LUMR approach and to originate from the small 'discontinuities' in the grid functions which are caused by the interpolation and injection (cf. the discussion of Section 7). A particular role hereby is played by the internal Dirichlet boundaries. The following experiment serves to illustrate this observation.

We have repeated the convergence test while omitting injection of fine grid results into coarse grid nodes. Hence, the only change in the algorithm is that coarse grid results are never updated so as to reduce the nonsmoothness in the grid functions. Note that this way for each level solutions are obtained only by integration or by interpolation. Further, the number of nodes where interpolation takes place will be relatively small since this is needed only at locations where the current level not yet exists. In this non-update convergence test we observed an overall improvement in accuracy. For instance, for increasing number of levels, the errors at $t = 0.25$ with their successive ratios now are, respectively, 0.21759, 0.04425, 0.01045, 0.00240, 0.00082 and 4.9, 4.2, 4.4, 2.9. Apparently, till level 4 the order has improved, but it drops again at level 5. Inspection of the output data revealed that in the level-5 run the maximal error was not always committed at this finest level-5 grid. This means that in the level-5 run the heuristic spatial error indicator (7.1) probably failed in adequately identifying the highest error region and this observation should explain the drop in order. If this, more or less technical, problem can be overcome, the 'non-update version' of the LUMR algorithm might be a remedy to the reduction in order and a good alternative for the more standard 'update version'. On the other hand, omitting updating may be somewhat dangerous because the coarse grid solution can become so dispersed that the algorithm decides that mesh refinement is no longer necessary. Of course, this can also happen when updating is used since we must rely here on error indicators or estimators. More precisely, since error indicators or estimators always underly some form of asymptotics, approximations serving as input for them should be of a certain minimal degree of accuracy in order to let them detect inaccuracies safely. For an LUMR method this implies that some care must be exercised in selecting the base grid parameters not too large since this might work out in the wrong way.

We conclude this numerical example with a measurement of overhead, in terms of total execution time and total number of PDE evaluations counted per node. The estimated part of the total execution time that is spent outside the integration routine is defined as overhead. Table 8.2 contains execution times and number of PDE evaluations for four runs over the time interval [0,0.25], using, respectively, 1, 2, 3 and 4 refinement levels. The data are obtained with the 'update' version using now also variable step sizes in time. The computations were organized such that at each run the finest grid has $\Delta x = \Delta y = 0.0125$. Hence, the level-1 run is performed on an 80-by-80 level-1 mesh with $\Delta x = \Delta y = 0.0125$, the level-2 run on a 40-by-40 level-1 mesh accompanied with a level-2 mesh with $\Delta x = \Delta y = 0.0125$, and so on. This way the maximal error committed during each of the four runs is more or less equal, so that the measurement of overhead is not

Levels	1	1-2	1-2-3	1-2-3-4	1-2-3-4-5
tols	4.0	1.0	0.25	0.125	0.03125
Δt	0.01	0.005	0.0025	0.00125	0.000625
Time point	Global errors measured in the maximum norm over the coarsest 10-by-10 grid. The numbers in the brackets are the corresponding error ratios.				
0.10	0.19120	0.03886 (4.9)	0.01154 (3.7)	0.00340 (3.4)	0.00112 (3.0)
0.15	0.24592	0.04904 (5.0)	0.01386 (3.5)	0.00420 (3.3)	0.00132 (3.2)
0.20	0.18575	0.03877 (4.8)	0.01150 (3.4)	0.00359 (3.2)	0.00119 (3.0)
0.25	0.21759	0.04511 (4.8)	0.01273 (3.5)	0.00383 (3.3)	0.00121 (3.2)

TABLE 8.1. Results of the convergence test on Problem I.

interfered by large differences in accuracy (in this reasoning we neglect the small effects of the above observed order-reduction phenomenon). The finest grid with $\Delta x = \Delta y = 0.0125$ is invoked by the spatial error indicator value tols = 0.125. The

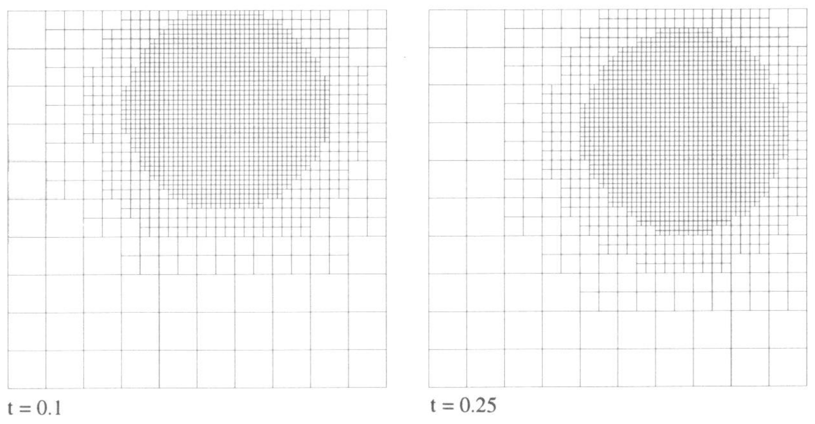


FIGURE 8.1. Grid structures used in the level-4 run with Problem I.

Levels	PDE evaluations	Execution times (sec)	Overhead %
1	9749646	865	0
1-2	3604712	365	12
1-2-3	3427610	381	20
1-2-3-4	4147352	512	28

TABLE 8.2. Results of time measurements for Problem I.

time step tolerance parameter tolt = 2.0E-7 and the initial time step equals 0.005. The parameter tolt was tuned so as to obtain nearly the same accuracy as in the level-4 run of the convergence experiment.

The entries 'overhead' in Table 8.2 are percentages defined by

$$\text{overhead} = \frac{((\text{time(level)} - (\text{eval(level)} * 865)/9749646)}{\text{time (level))}} * 100\%, \qquad (8.4)$$

where eval(level) is the total number of PDE evaluations and time(level) is the execution time. Thus in our measurement of overhead the work load for the single, 80-by-80 mesh is used as a reference point. When inspecting Table 8.2 one should realize that the overhead factors are valid only for this particular experiment. Overhead is not only solution dependent (size of the refined grids), but, since we measure CPU times, also depends on the differential equation (expensive expressions in the operator) and on the degree of optimality in the FORTRAN code for the data structure and the like. As yet, we have paid little attention to the matter of optimal coding. In our opinion, the overhead factors found in this experiment are quite acceptable, although we should note that the use of garbage collection will increase the overhead (cf. Section 4). Finally we wish to note that for the present experiment the decrease in execution time shown in Table 8.2 is minor due to the fact that the refined grids are still of considerable size.

2.8.2. Problem II.

Our second example problem has also been borrowed from [1] and stems from combustion theory. The problem is a model for a so-called single, one-step reaction of a mixture of two chemicals. In the problem, the dependent variable u represents the temperature of the mixture. The equation reads

$$u_t = d(u_{xx} + u_{yy}) + D(1+\alpha-u)\exp(-\frac{\delta}{u}), \qquad 0 < x,y < 1, \qquad t>0, \qquad (8.5)$$

and is subjected to the following initial and boundary conditions,

$$\begin{aligned} u(x,y,0) &= 1, \quad 0 \le x,y \le 1, && (8.6)\\ u_x(0,y,t) &= 0, \quad u(1,y,t) = 1, \quad 0 \le y \le 1, \quad t > 0, \\ u_y(x,0,t) &= 0, \quad u(x,1,t) = 1, \quad 0 \le x \le 1, \quad t > 0. \end{aligned}$$

The parameter α is the heat release, $D = R\exp(\delta)/\alpha\delta$ the Damkohler number, δ the activation energy, and R is the reaction rate. For small times the temperature gradually increases in a circular area around the origin. Then, provided the reaction rate is large enough, at a finite time 'ignition' occurs causing the temperature to suddenly jump from near unity to $1+\alpha$, while simultaneously a reaction front is formed which circularly propagates towards the outer Dirichlet boundary. When the front reaches the boundary the problem runs into steady state. Following [1] we select the parameter values $\alpha = 1$, $\delta = 20$, $R = 5$, but choose a different value for the diffusion parameter. While in [1] the diffusion coefficient $d = 1.0$, we here put $d = 0.1$. A smaller diffusion coefficient has the effect that the wave front becomes steeper, particularly so upon approaching steady state [14]. With this choice of parameters the 'ignition' takes place at about $t = 0.24$ and the solution is in 'steady state' at about $t = 0.35$.

In spite of the fact that a fine grid is necessary for combustion problems of this type, the explicit RKC scheme is a natural candidate for the numerical integration. Two arguments support this observation. First, we must follow a traveling front on static, i.e., non-moving space grids. This naturally limits the temporal step size of any integration scheme, be it explicit or implicit. Second, during the time evolution this special combustion problem is 'locally unstable'. Inspection of the reaction term reveals that for $1 \leq u \leq 2$ its derivative varies approximately between +1000 (for $u \approx 1.6$) and -5500 (for $u \approx 2.0$). Consequently, irrespective the integrator used, quite small integration steps are required to maintain sufficient accuracy in regions of 'local instability' and before steady state will be reached the advantage of unconditional stability as provided by implicit methods shall not be fully exploited. To our experience, for this problem the explicit, stabilized RKC method is a good alternative.

Figure 8.2 shows generated grids and solutions at six specified time points. These are obtained using the full variable step size in time option using a uniform base grid with $\Delta x = \Delta y = 0.05$, tols = 0.6, tolt = 1.0E-7 and initial step size of 0.005. With this choice of parameters the method uses at most three levels, but integrates till $t = 0.24175$ only on the coarse base grid. A notable point is that the grids accurately reflect the symmetrical shape of the combustion wave front, thus again showing that our regridding strategy works very well. The steepening up of the wave front for evolving time is also clearly visible from the width of the finest level-3 grid. The solution at the end time $t = 0.35$ is, approximately, in steady state and shows a thin layer at the outer Dirichlet boundaries. At this time point the level-3 grid near the layer is only 6 cells wide, which is the minimum number that is possible near a boundary.

Inherent in static-regridding methods is that they have some difficulty in efficiently approaching steady state. When using a single grid without refinement, and assuming stability, a good variable step size solver will steadily increase the step size upon approaching steady state. On fine level grids this steady increase of step size will be less for a static-regridding method like ours, at least when using the 'update version', due to the fact that the injection of solutions from fine grids into coarse ones has the effect that the numerical solution at the coarse grids is 'slightly perturbed'. The 'slight perturbations' hinder the approximation from steadily becoming stationary like in single-grid computations. This negative effect will be most pronounced at higher levels of refinement. Fortunately, the drawback can be largely overcome by using the 'non-update version' where injection is omitted.

By way of illustration we have included Table 8.3. This table contains information concerning the temporal integration for three runs:

(i) the run corresponding with Figure 8.2, where 'updating' was used,
(ii) a similar run, but now without 'updating',
(iii) a run using a single, 80-by-80 grid without refinement, with the same parameter values for the variable step size control.

We note that in case (ii) the generated grids are nearly the same as in case (i) and that the plot accuracy in the three runs is the same. In particular, the finest mesh width in space for all three runs is $\Delta x = \Delta y = 0.0125$, so that, since the accuracy is very much the same, we can compare the workload. We see that for (i) and (ii) the workload is nearly the same; (ii) requires more steps at level 1, but less at level 2 and 3 as to be expected. The number of time steps in (iii) is nearly the same as at

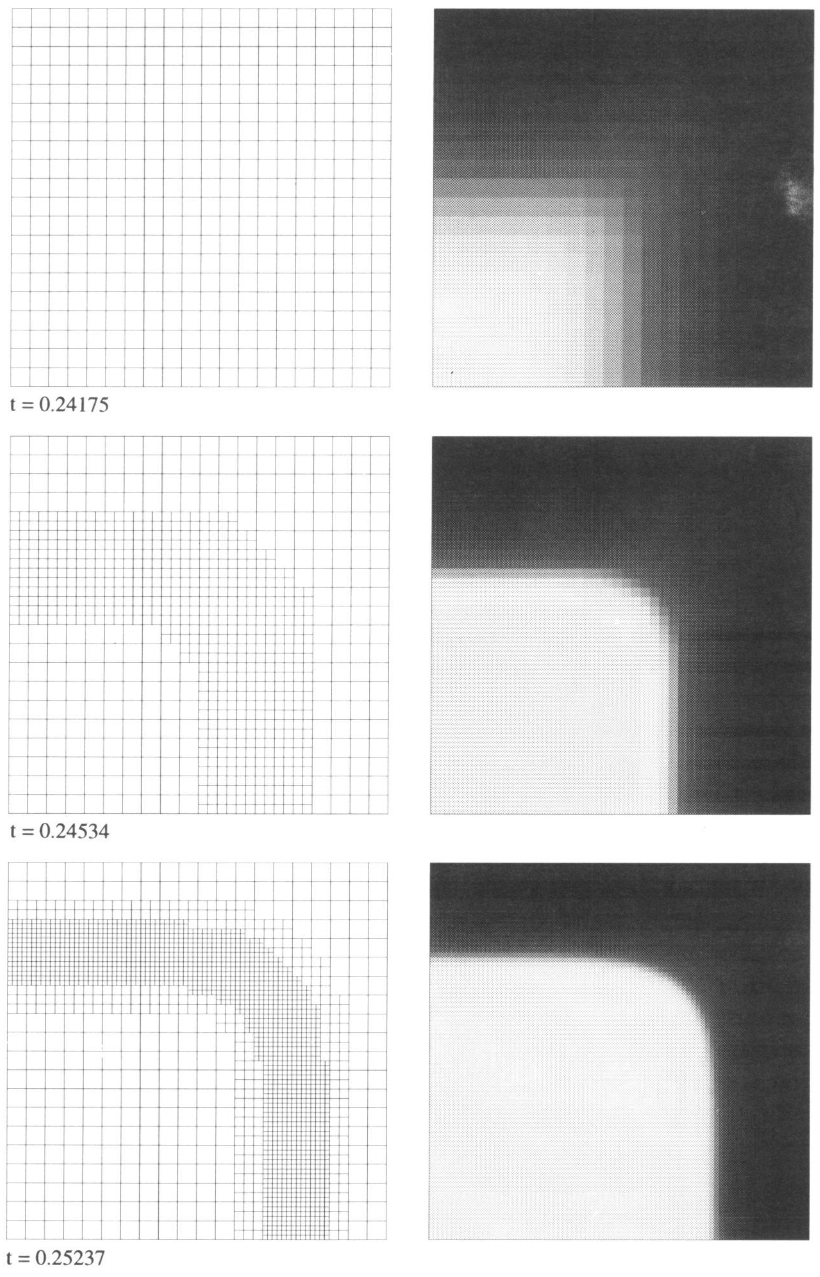
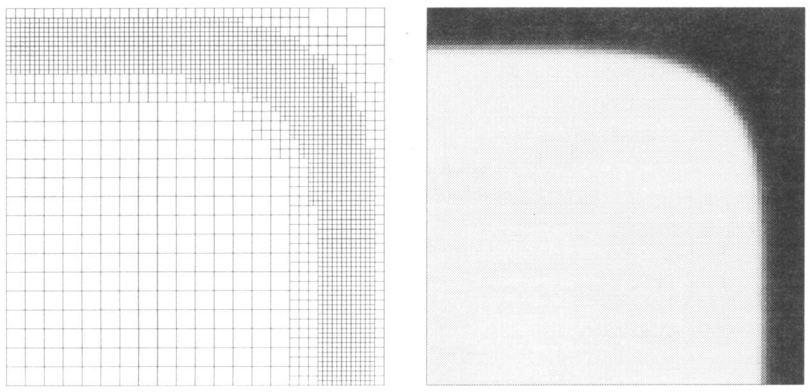
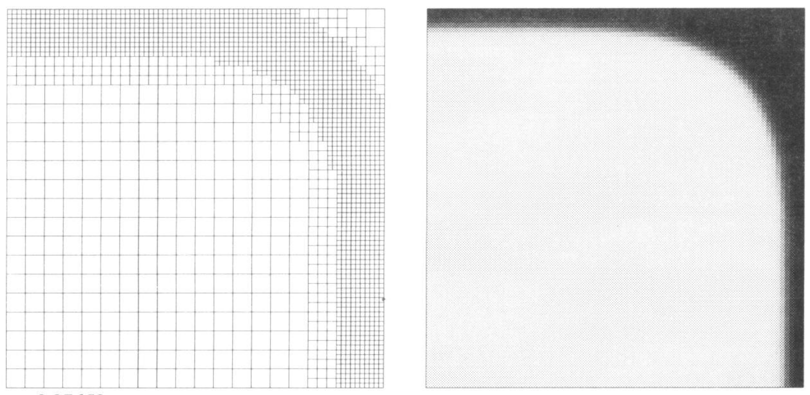


FIGURE 8.2. Grids and solutions of Problem II at some specified output points.


t = 0.26796


t = 0.27653

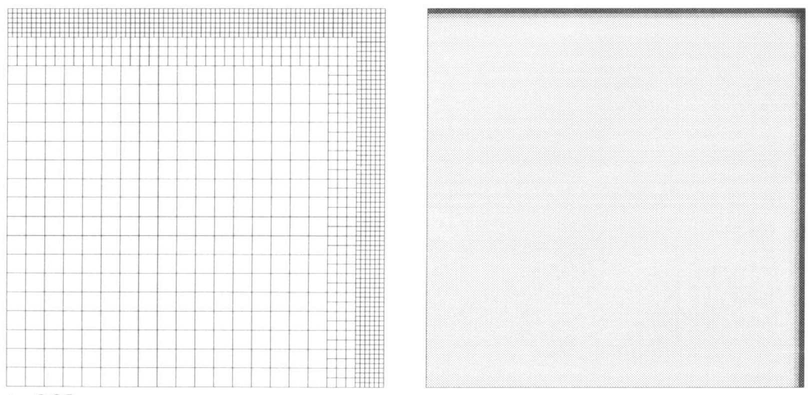

t = 0.35

FIGURE 8.2. Continued.

level 1 in (i) and (ii). The reduction in execution time, when comparing (i) and (ii) with (iii) is approximately a factor of 2.7. This includes CPU time for overhead which here appears to be larger than for Problem I.

	Accepted time steps			PDE	Execution	Overhead
	level 1	level 2	level 3	evaluations	times (sec)	%
(i)	186	193	178	1530249	264	37
(ii)	199	168	154	1470343	252	37
(iii)	195	0	0	6206706	671	0

TABLE 8.3. Results of time measurements for Problem II. The number of rejected steps is negligible in all three cases. The overhead is defined in the same way as for Problem I, now relative to experiment (iii).

2.9. FINAL REMARKS AND FUTURE PLANS.

The LUMR algorithm presented in this paper is based on a mix of various techniques and builts on previous work started by *Berger* and *Oliger* [4] and continued by *Gropp* [9-11], *Arney* and *Flaherty* [2] and others. While *Berger* and *Oliger* [4] consider hyperbolic problems, we have focussed on parabolic problems and have applied a special, explicit time integrator using variable time steps. The integrator is special in that it possesses an extended real stability interval. For parabolic problems of type (1.1), the integrator is therefore an attractive alternative in situations where the good stability properties of the more common implicit solvers can not be fully exploited due to inherent step size restrictions. Our example Problem II illustrates such a situation.

Further, while *Berger* and *Oliger* [4] and *Arney* and *Flaherty* [2] use noncellular refinement and truly rectangular subgrids which may overlap and rotate to align with an evolving dynamic structure, we have chosen to use cellular refinement and to avoid overlapping subgrids. On the other hand, our subgrids are not necessarily truly rectangular and may of course be disjunct with neighboring subgrids at the same level of refinement. This approach allows a simpler data structure and, most importantly, makes it possible in a relatively simple and transparent way to exclusively interpolate missing initial and boundary conditions at internal grid interfaces in low-error regions. It is emphasized that our method, unlike the method of *Berger* and *Oliger* [4] due to overlapping subgrids, does not allow steep solutions to intersect grid interfaces. For this purpose, a reliable refinement strategy is of crucial importance. A good refinement strategy should refine in such a way that the accuracy obtained at the current highest level grid is comparable to the accuracy obtained on this grid if it would be used without any adaptation. This way the interpolation errors will never become visible. Our refiment strategy, as discussed in

Section 3, attempts to achieve this through the use of the refinement condition (3.6). This condition in fact looks ahead such that all cells at the current level are refined, except those for which at the anticipated highest level the solution is already sufficiently accurate. The rationale behind this strategy is that when at a certain level a cell is not refined, we never return to this cell within the current base time step as we work with nested subgrids.

In connection with the use of nested subgrids, we recall that each time step we restart from the base grid, but also that we keep the finest grid solution in storage for possible use in the next time step. Actually, for evolving time we integrate on different grid levels with the understanding that the integration domains are nested per level and change in time. The nesting requires the interpolation of Dirichlet boundary values and the change in time requires interpolation of initial values, but only for those nodal points not already used at the previous time step. Most of the time we thus restart from the finest grid solution that already exist at a given nodal point.

The advantage of this approach, which is typical for LUMR methods, is that one can integrate on uniform subgrids and avoid the use of truly nonuniform grids such as obtained in a pointwise refinement procedure (see, e.g., *Adjerid* and *Flaherty* [1]). Admittedly, the approach may be a bit wasteful in situations where the sharp transitions move very slowly, e.g., when approaching steady state. On the other hand, the computational effort for the coarser grids normally shall not be large and finite-difference or finite-element expressions on uniform grids are more accurate and cheaper to process than on nonuniform grids. Also note that an inherent drawback of pointwise refinement and a truly nonuniform grid is a more complex and expensive data structure.

There are two major reasons why the development of LUMR methods is of interest. The first reason is obvious and of a purely practical nature: by refining the spatial grid locally in regions of high spatial activity, it is attempted to obtain accurate numerical solutions at significantly lower costs than required in the standard approach without refinement. Our second numerical example illustrates this nicely in respect with the execution time (see Table 8.3). Of course, since any LUMR method necessarily involves considerable overhead arising from the data structure, the regridding, the repeated integrations, etc., these methods are of interest only when the spatial solution variations are sufficiently large, like in our Problem II. We also conclude that in both our numerical examples the actual regridding strategy based on the refinement condition (3.6) has functioned very well. This follows directly from inspection of the plotted grids. On the other hand, in this paper the actual input for the regridding still stems from the heuristic indicator (7.1). Our choices of tols illustrate very clearly that this 'curvature indicator' bears no good resemblance with the true spatial errors. No doubt it will be very worthwhile to replace (7.1) by an accurate estimator of the genuine local space error, assuming this is feasible. This of course is also true for the time error indicator (7.2).

These observations lead us to the second reason why LUMR methods are of interest. This second reason is of a more fundamental nature: these methods offer the natural environment for balancing genuine local space and time errors and to let the local refinement be governed by the genuine local space error. Balancing local space and time errors has so far got only very little attention in the literature, but is of obvious importance when one aims at efficiency and robustness in solving time-dependent PDEs (see *Berzins* [5] and *Schönauer* et al. [18]). We will therefore continue our work on adaptive grid methods with an investigation to convergence properties of the present LUMR method. In this investigation we plan to analyze both implicit and explicit methods and an important goal will be balancing genuine local time and space errors. In this connection it should be noted that L-stable implicit Runge-Kutta methods are of interest because of their inherent smoothing properties. This will no doubt help to reduce any difficulty originating from the unavoidable 'nonsmoothness' in the numerical solution.

ACKNOWLEDGEMENT

We would like to thank Joke Blom and Paul Zegeling for reading the manuscript and for suggesting many corrections and clarifying remarks.

REFERENCES

1. S. ADJERID and J.E. FLAHERTY (1988). A local Refinement Finite Element Method for Two Dimensional Parabolic Systems, *SIAM J. Sci. Statist. Comput.*, 9, 792-811.
2. D.C. ARNEY and J.E. FLAHERTY (1989). An Adaptive Local Mesh Refinement Method for Time-Dependent Partial Differential Equations, *Appl. Numer. Math.*, 5, 257-274.
3. M.J. BERGER (1986). Data Structures for Adaptive Grid Generation, *SIAM J. Sci. Statist. Comput.*, 7, 904-916.
4. M.J. BERGER and J. OLIGER (1984). Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, *J. Comput. Phys.*, 53, 484-512.
5. M. BERZINS (1988). Global Error Estimation in the Method of Lines for Parabolic Equations, *SIAM J. Sci. Statist. Comput.*, 9, 687-703.
6. I.S. DUFF, A.M. ERISMAN, and J.K. REID (1986). *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford.
7. R.E. EWING (1989). Adaptive Grid Refinement for Transient Flow Problems, in *Adaptive Methods for Partial Differential Equations*, 194-205, ed. J.E. FLAHERTY, P.J. PASLOW, M.S. SHEPHARD, J.D. VASILAKIS, SIAM Publications, Philadelphia.
8. J.E. FLAHERTY, P.J. PASLOW, M.S. SHEPHARD, and J.D. VASILAKIS, EDITORS. (1989). *Adaptive Methods for Partial Differential Equations*, SIAM Publications, Philadelphia.
9. W.D. GROPP (1980). A Test of Moving Mesh Refinement for 2D-Scalar Hyperbolic Problems, *SIAM J. Sci. Statist. Comput.*, 1, 191-197.
10. W.D. GROPP (1987). Local Uniform Mesh Refinement with Moving Grids, *SIAM J. Sci. Statist. Comput.*, 8, 292-304.
11. W.D. GROPP (1987). Local Uniform Mesh Refinement on Vector and Parallel Processors, in *Large Scale Scientific Computing*, 349-367, ed. P. DEUFLHARD, B.

ENGQUIST, Birkhäuser Series Progress in Scientific Computing.

12. A.C. HINDMARSH and S.P. NØRSETT (1988). *KRYSI, An ODE Solver Combining a Semi-Implicit Runge-Kutta Method and a Preconditioned Krylov Method,* Rept. UCID-21422, Lawrence Livermore National Laboratory.
13. P.J. VAN DER HOUWEN and B.P. SOMMEIJER (1980). On The Internal Stability of Explicit m-Stage Runge-Kutta Methods for Large m-Values, *Z. Angew. Math. Mech.*, 60, 479-485.
14. K. MILLER (1988). *Private Communication.*
15. L.R. PETZOLD (1987). *Adaptive Moving Grid Strategies for One-Dimensional Systems of Partial Differential Equations,* Preprint UCRL-96190, Lawrence Livermore National Laboratory.
16. L.R. PETZOLD (1987). Observations on an Adaptive Moving Grid Method for One-Dimensional Systems of Partial Differential Equations, *Appl. Numer. Math.*, 3, 347-360.
17. J.M SANZ-SERNA, J.G. VERWER, and W.H. HUNDSDORFER (1987). Convergence and Order Reduction of Runge-Kutta Schemes Applied to Evolutionary Problems in Partial Differential Equations, *Numer. Math.*, 50, 405-418.
18. W. SCHÖNAUER, E. SCHNEPF, and K. RAITH (1984). Numerical Engineering: Experiences in Designing PDE Software with Self Adaptive Variable Stepsize/Variable Order Difference Methods, *Computing Suppl.*, 5, 227-242.
19. B.P. SOMMEIJER (1989). *Private Communication.*
20. J.G. VERWER, W.H. HUNDSDORFER, and B.P. SOMMEIJER (1990). Convergence Properties of the Runge-Kutta-Chebyshev Method, *Numer. Math.*, 57, 157-178.
21. J.G. VERWER, J.M SANZ-SERNA, and J.G. BLOM (1989). An Adaptive Moving-Grid Method for One-Dimensional Systems of Partial Differential Equations, *J. Comput. Phys.*, 82, 454-486.

Chapter 3

Analysis of the Implicit Euler Local Uniform Grid Refinement Method

R.A. Trompert and J.G. Verwer
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

Attention is focussed on parabolic problems having solutions with sharp moving transitions in space and time. An adaptive grid method is analyzed that refines the space grid locally around sharp spatial transitions, so as to avoid discretization on a very fine grid over the entire physical domain. This method is based on static regridding and local uniform grid refinement. Static regridding means that for evolving time the space grid is adapted at discrete times. Local uniform grid refinement means that the actual adaptation of the space grid takes place using nested, local, uniform grids. The present paper concentrates on stability and error analysis while using the implicit Euler method for time integration. Maximum norm stability and convergence results are proved for a certain class of linear and nonlinear partial differential equations. The central issue is a refinement condition with a strategy that distributes spatial interpolation and discretization errors in such a way that the spatial accuracy obtained is comparable to the spatial accuracy on the finest grid if this grid would be used without any adaptation. The analysis is confirmed with a numerical illustration.

published in: SIAM J. Sci. Comput. **14** (2) (1993), 259-278

3.1. Introduction

Attention is focussed on parabolic problems having solutions with sharp moving transitions in space and time, such as steep fronts and disappearing layers. For such problems, a space grid held fixed throughout the entire time evolution can be computationally very inefficient. We consider an adaptive grid method that refines locally around sharp spatial transitions so as to avoid discretization on a very fine grid over the entire physical domain.

Our method is based on the techniques called static regridding and local uniform grid refinement (LUGR), as previously proposed by *Berger* and *Oliger* [3], *Gropp* [6-8], *Arney* and *Flaherty* [2], *Flaherty*, *Moore* and *Ozturan* [11], *Trompert* and *Verwer* [13], and others. Static regridding means that for evolving time the space grid is adapted at discrete times. This should be contrasted with dynamic regridding where the space grid moves continuously in the space-time domain. With the term LUGR we mean that the actual adaptation of the space grid takes place using local, uniform, refined grids. LUGR should be contrasted with pointwise refinement which leads to truly nonuniform grids. In this connection, our LUGR method bears resemblance with the fast adaptive composite grid (FAC) method [10] for elliptic equations, where the basic computational objective is to solve on an irregular grid by way of regular grids only.

The idea of the method can be briefly described as follows. Given a coarse base grid and a temporal step size, nested, local, uniform subgrids are generated. These subgrids possess nonphysical boundaries and on each of these subgrids an integration is carried out. They are generated up to a level of refinement good enough to resolve the anticipated fine scale structures. Having completed the refinement for the current base space-time grid, the process is continued to the next one while the fine grid results computed at forward time levels are kept in storage so that they can be used for step continuation.

An attractive feature of the static-regridding approach is the possibility of dividing the solution process into the following computational procedures: spatial discretization, temporal integration, error estimation, regridding and interpolation. Depending on the application, these individual procedures may range from simple or straightforward to very sophisticated. This flexibility is attractive since it makes it possible to treat different types of partial differential equations (PDEs) with almost one and the same code, assuming that the grid and the associated data structure remain unchanged. Note that the choice of data structure is important for keeping the unavoidable overhead at an acceptable level, because at each time step grids may be created or removed while also communication between grids of adjacent levels of refinement frequently takes place.

The method we analyze in this paper has many similarities with the method constructed in *Trompert* and *Verwer* [13]. In fact, the grid and data structure, the spatial differencing and the memory use are the same. However, in the present paper we concentrate on analysis rather than on construction, while using implicit Euler instead of the explicit Runge-Kutta-Chebyschev method for time integration. The main aim of this paper is to present a detailed error analysis and to prove stability and convergence for a certain class of PDEs. The central issue in this analysis is a refinement condition and a strategy that distributes spatial discretization and interpolation errors in such a way that the spatial accuracy obtained is comparable to the spatial accuracy on the finest grid if this grid would be used without any adaptation.

Section 2 is devoted to the problem class we concentrate on. In Section 3 we introduce the tools and the formulation for the multilevel LUGR method. In Section 4 we discuss the maximum-norm stability of this method. We prove an unconditional stability result which is closely related to a maximum-norm stability result of implicit Euler when applied on a single space grid. Section 5 is devoted to the error analysis. In this section we investigate the total local error with its component parts. Furthermore, here we introduce the refinement strategy underlying the so-called refinement condition. This condition enables us to control the contribution of the interpolation errors in favor of discretization errors. Due to this condition, we are able to prove a convergence result as if we are working on a single, fixed grid. We further elaborate on this condition in Section 6 where we show how to implement it for practical use. A numerical illustration of the error analysis is given in Section 7. The numerical results found here are in complete agreement with the analysis. Finally, Section 8 briefly discusses our future research plans.

3.2. The problem class

Following the method-of-lines approach [12], we consider a real abstract Cauchy problem

$$u_t = L(t, u), \qquad 0 < t \leq T, \qquad u(\underline{x}, 0) = u^0(\underline{x}), \tag{2.1}$$

where L represents a second-order partial differential operator which differentiates the (possibly vector valued) solution $u(\underline{x},t)$ to its space variable $\underline{x}$ in a space domain Ω in $\mathbb{R}$, $\mathbb{R}^2$ or $\mathbb{R}^3$. Boundary conditions are supposed to be included in the definition of L.

With (2.1) we associate a real Cauchy problem for an explicit system of ordinary differential equations (ODEs) in $\mathbb{R}^d$,

$$\frac{d}{dt}U(t) = F(t, U(t)), \qquad 0 < t \leq T, \qquad U(0) = U^0, \tag{2.2}$$

which is defined by a finite-difference space-discretization. Thus, U and F are vectors in $\mathbb{R}^d$ representing grid functions on a space grid ω covering the interior of the space domain. Each component of U and F is vector valued if u is vector valued. The dimension d is determined by the spatial dimension, the grid spacing, and the number of PDEs in (2.1). F is determined by the type of grid, by the actual finite-difference formulas and of course by the precise form of L and its boundary conditions. Note that boundary values have been eliminated and worked into the ODE system. In the following, our method description and analysis are centered around this system.

Next we introduce some notations and assumptions needed for further specifying (2.1) and (2.2). The symbol $\|.\|$ denotes the maximum norm on the vector space $\mathbb{R}^d$ or the induced matrix norm. Throughout our analysis we will deal only with the maximum norm. The symbol $\mu[A]$ denotes the logarithmic matrix norm of the real $d \times d$ matrix $A = (a_{ij})$ associated with $\|.\|$, i.e.,

$$\mu[A] = \max_i \{(a_{ii} + \textstyle\sum_{j \neq i} |a_{ij}|)\}. \tag{2.3}$$

$\mu[A]$ is a useful tool in the stability analysis of nonlinear, stiff ODEs and semi-discrete PDEs [4]. In this analysis, the structure of the Jacobian matrix $F'(t,\eta) = \partial F(t,\eta)/\partial\eta$ plays a decisive role.

We are now ready to list our assumptions we make in further specifying (2.1) - (2.2). These assumptions are concerned with, respectively, the class of PDEs (2.1), the smoothness of u, the choice of spatial grid and actual finite-differencing, and the stability of the semi-discrete system (2.2):

(A1) The LUGR method is applicable in any number of space dimensions. Following [13], we concentrate on the 2D case, while Ω is supposed to be the unit square. With minor changes Ω is allowed to be composed of a union of rectangles with sides parallel to the co-ordinate axes. In fact, as we will see later, refined grids normally are of this shape. In what follows, we will mostly use the notation $u(x,y,t)$, rather than $u(\underline{x},t)$.

(A2) The solution u of (2.1) uniquely exists and is as smooth as the numerical analysis requires. Specifically, for our purpose it suffices that, u is a C^2-function in t and a C^4-function in (x,y).

(A3) We will invariably use uniform space grids. Our base grid thus can be written as

$$\omega = \{(x_i,\ y_j)\colon x_i = ih_x,\ \ 1 \le i \le M-1 \text{ and } y_j = jh_y,\ \ 1 \le j \le N-1\}, \tag{2.4}$$

where $h_x = 1/M$, $h_y = 1/N$ and M, N are positive integers. The spatial differencing on ω is supposed to be based on three-point formulas of second-order consistency. As a rule, we use central differencing conditions involving first-order derivatives, the one-sided three-point formula is used.

(A4) A constant ν exists such that $\mu[F'(t,\eta)] \le \nu$ for all $t \in (0,T]$, $\eta \in \mathbb{R}^d$ and all grid spacings. Like (A1) and (A2), this assumption involves a restriction on the class of PDE problems. Of course, they are made only for the sake of (model) analysis. The LUGR method remains applicable in situations where these assumptions do not hold or cannot be verified. On the other hand, for interesting classes of operators, such as the scalar, nonlinear parabolic operator

$$L(t,u) = f_1(t,x,y,u,(p_1(t,x,y)u_x)_x) + f_2(t,x,y,u,(p_2(t,x,y)u_y)_y), \tag{2.5}$$

with standard restrictions on f_i and p_i, one can prove the existence of a constant ν.

The inequality $\mu[F'(t,\eta)] \le \nu$ is to be interpreted as a stability condition, both concerning the ODE system (2.2) and its implicit Euler discretization

$$U^n = U^{n-1} + \tau F(t_n,\ U^n), \qquad n = 1,2,...., \tag{2.6}$$

where $\tau = t_n - t_{n-1}$ is the step size and U^n is the approximation for $U(t_n)$. This inequality enables us to formulate the following, powerful stability result for implicit

Euler. Consider the perturbed form

$$\tilde{U}^n = \tilde{U}^{n-1} + \tau F(t_n, \tilde{U}^n) + r^n, \qquad n = 1,2,...., \tag{2.7}$$

where r^n is an arbitrary local perturbation and $\tilde{U}^{n-1}$, $\tilde{U}^n$ are perturbations to U^{n-1}, U^n. Then

$$\|\tilde{U}^n - U^n\| \leq \frac{1}{1-\tau\nu} \|\tilde{U}^{n-1} - U^{n-1} + r^n\|, \qquad n = 1,2,...., \tag{2.8}$$

for all $\tau > 0$ satisfying $\tau\nu < 1$ [4]. Since ν is independent of the grid spacing, this stability inequality is valid uniformly in h_x and h_y. For $\nu = 0$ we have contractivity for all $\tau > 0$, while for $\nu < 0$ we even have damping for all $\tau > 0$. A result closely related to (2.8) will be derived in Section 4.

3.3. The implicit Euler local uniform grid refinement method

3.3.1. Outline

Although its elaboration readily becomes complicated, the idea behind LUGR is simple. Starting from the coarse base grid covering the whole domain, finer-and-finer uniform subgrids are created locally in a nested manner in regions of high spatial activity. These subgrids are created by bisecting sides of next coarser grid cells. A new initial-boundary value problem is solved at each subgrid, and the integration takes place in a consecutive order, from coarse to fine. Each of these integrations spans the same time interval. Required initial values are defined by interpolation from the next coarser subgrid or taken from a subgrid from the previous time step when available. Internal boundaries are treated as Dirichlet boundaries and values are also interpolated from the next coarser subgrid. The generation of subgrids is determined by the local refinement strategy and is continued until the spatial phenomena are described well enough by the finest grid.

During each time step the following operations are performed:

(1) *Integrate on coarse base grid.*
(2) *Determine new finer uniform subgrid at forward time.*
(3) *Interpolate internal boundary values at forward time.*
(4) *Provide new initial values at backward time.*
(5) *Integrate on subgrid, using the same step length.*
(6) *If the desired accuracy in space is reached go to 7, else go to 2.*
(7) *Inject fine grid values in coinciding coarser grid points.*

Thus, for each time step, the computation starts at the coarse base grid using the most accurate solution available, since fine solution values are always injected in coinciding coarse grid points. Moreover, all subgrids are kept in storage for step continuation.

We consider the use of uniform grids attractive because uniform grids allow an efficient use of vector-based algorithms and finite differences on uniform grids are faster and more accurate to compute than on nonuniform grids. In this respect the current approach is to be contrasted with pointwise refinement leading to truly nonuniform grids. Pointwise refinement techniques also require a more involved data structure (*Ewing* [5]). On the other hand, with the LUGR method, there are nodes that exist on more than one grid at the same time, meaning that at these nodes integration takes place more than once during one time step. Hence, the total number of nodal integrations needed will be larger than on a comparable single, nonuniform grid.

In [2, 3, 7, 11] LUGR methods are examined based on noncellular refinement and truly rectangular subgrids which may rotate and overlap to align with an evolving fine scale structure. We avoid these difficulties. Our local subgrids do not overlap, they may be disjunct, they need not be rectangles, and the actual refinement is cellular.

3.3.2. The mathematical formulation

LUGR methods solve PDEs on the whole domain at the coarsest grid only and on a part of the domain at finer subgrids. Our method can be interpreted as a sequence of operations on vectors in $\mathbb{R}^d$ with varying dimension d. The dimensions are time and level dependent because the number of nodes changes per level of refinement and per time step. This constitutes a problem for the formulation of the method. To bypass this difficulty, the fine grids will be expanded so that they cover the whole domain. The dimensions are then fixed per level of refinement, which facilitates the derivation of a concise mathematical formulation. We emphasize that this grid expansion is auxiliary. In actual application only part of the expanded higher-level grids is processed.

Suppose that for a given time interval $[0,T]$ and a given base grid, l levels are needed to describe the spatial activity of a solution sufficiently accurately when integrating over the entire time interval $[0,T]$. Introduce for $k = 1, \cdots, l$ the expanded uniform grids

$$\begin{aligned}\omega_k = \{(x_i,\ y_j)\colon x_i = ih_{x,k},\ & 1 \le i \le 2^{k-1}M-1 \text{ and} \\ y_j = jh_{y,k},\ & 1 \le j \le 2^{k-1}N-1\},\end{aligned} \tag{3.1}$$

where N and M are the same integers as in (2.4) and $h_{x,k} = h_x/2^{k-1}$, $h_{y,k} = h_y/2^{k-1}$. Note that for $k = 1$ the base grid $\omega_1 = \omega$ given by (2.4) is recovered.

Let the generic notation for a grid function η defined at ω_k be η_k and let S_k denote the space of these grid functions. We then denote the semi-discrete system considered in S_k by

$$\frac{d}{dt}U_k(t) = F_k(t, U_k(t)), \qquad 0 < t \le T, \qquad U_k(0) = U_k^0. \tag{3.2}$$

Note that due to the grid expansion, only a part of the components of the ODE system (3.2) is integrated for $k > 1$ in reality.

We are now ready to formulate the implicit Euler LUGR method. The following formula defines the time step from step point t_{n-1} to t_n for l levels of refinement:

$$U_1^n = R_{l1}U_l^{n-1} + \tau F_1(t_n, U_1^n), \tag{3.3a}$$

$$U_k^n = D_k^n\,[R_{lk}U_l^{n-1} + \tau F_k(t_n, U_k^n)] + (I_k - D_k^n)\,[P_{k-1k}U_{k-1}^n + b_k^n], \tag{3.3b}$$

for $k=2,\ldots.,l$, where

$U_k^n \in S_k^n$ is the approximation to u at ω_k at $t = t_n$,
I_k: $S_k \rightarrow S_k$ is the unit matrix,
D_k^n: $S_k \rightarrow S_k$ is a diagonal matrix with entries $(D_k^n)_{ii}$ either unity or zero,
R_{lk}: $S_l \rightarrow S_k$ is the natural restriction operator from ω_l to ω_k
P_{k-1k}: $S_{k-1} \rightarrow S_k$ is an interpolation operator from ω_{k-1} to ω_k
$b_k^n \in S_k$ contains time-dependent terms emanating from the boundary $\partial\Omega$.

Specifically, the nonzero entries of D_k^n $(2 \le k \le l)$ are meant to determine that part of ω_k where the actual integration takes place. This integration has the fine grid solution $D_k^n R_{lk} U_l^{n-1}$ as initial function and is defined by

$$D_k^n U_k^n = D_k^n\,[R_{lk}U_l^{n-1} + \tau F_k(t_n, U_k^n)], \qquad k=2,\ldots.,l. \tag{3.4}$$

The definition of D_k^n is provided by the refinement strategy. For the time being there is no need to further specifying D_k^n. Note that the nesting property of the integration domains is hidden in the precise definition of the matrices D_k^n. The interpolation step is defined by

$$(I_k - D_k^n)U_k^n = (I_k - D_k^n)\,[P_{k-1k}U_{k-1}^n + b_k^n], \qquad k=2,\ldots.,l, \tag{3.5}$$

where the grid function b_k^n contains various time-dependent terms occurring in physical boundary conditions. We need to include b_k^n because physical boundary conditions have been worked into the semi-discrete system. For the analysis in the remainder, b_k^n plays no role whatsoever.

The formulation (3.3) automatically comprises the interpolation of boundary values at grid interfaces. This follows directly from the observation that for nodes at grid interfaces, the associated diagonal entry of D_k^n is zero (there is no integration at grid interfaces). Further, we note that (3.3) implies an order, (3.3a) is carried out for the coarse base grid and (3.3b) for $k=2,\ldots.,l$ successively. Having done this, the updating will take place, meaning that U_k^n is replaced by $R_{lk}U_l^n$ from $k=l-1$ to 1. After this we move on to the next time step. Recall that, due to the grid expansion, in (3.3) the interpolation is carried out for all nodal points outside the integration domain of ω_k. This enables the stability and convergence analysis to be carried out for the spaces S_k. However, in actual application interpolation only takes place at the local subgrids. In Section 6.2 it is shown that this does not interfere with the analysis.

3.4. Stability analysis

3.4.1. Preliminaries

Consider, on the analogy of (2.7), for $n=1,2,....$, the perturbed scheme

$$\tilde{U}_1^n = R_{l1}\tilde{U}_1^{n-1} + \tau F_1(t_n, \tilde{U}_1^n) + r_1^n, \tag{4.1a}$$

$$\begin{aligned}\tilde{U}_k^n = D_k^n\,[R_{lk}\tilde{U}_l^{n-1} + \tau F_k(t_n, \tilde{U}_k^n)] + \\ (I_k - D_k^n)\,[P_{k-1k}\tilde{U}_{k-1}^n + b_k^n] + r_k^n,\end{aligned} \tag{4.1b}$$

for $k=2,....,l$ with local perturbations r_k^n, and introduce the errors $e_k^n = \tilde{U}_k^n - U_k^n$, for $k=1,....,l$. To shorten the formulas, we introduce the auxiliary quantities e_0^n, D_1^n and P_{01}, where $e_0^n = 0 \in S_1$, D_1^n is the unit matrix I_1, and P_{01} is the zero matrix. Then, by subtracting (3.3) from (4.1), we get

$$Z_k^n e_k^n = D_k^n R_{lk} e_l^{n-1} + (I_k - D_k^n)P_{k-1k}e_{k-1}^n + r_k^n, \tag{4.2}$$
$$n=1,2,.... ; \quad k=1,....,l,$$

where $Z_k^n = I_k - \tau D_k^n M_k^n$ and is M_k^n the integrated Jacobian matrix

$$M_k^n = \int_0^1 F'(t_n, \theta\tilde{U}_k^n + (1-\theta)U_k^n)d\theta \tag{4.3}$$

which results from applying the mean value theorem for vector functions.

Assuming Z_k^n to be nonsingular, we can rewrite (4.2) as

$$e_k^n = \mathrm{X}_k^n e_{k-1}^n + \Gamma_k^n e_l^{n-1} + \phi_k^n, \quad n=1,2,.... ; \quad k=1,....,l, \tag{4.4}$$

with

$$\begin{aligned} X_k^n &= (Z_k^n)^{-1}(I_k - D_k^n)P_{k-1k}, \\ \Gamma_k^n &= (Z_k^n)^{-1}D_k^n R_{lk}, \\ \phi_k^n &= (Z_k^n)^{-1} r_k^n. \end{aligned} \tag{4.5}$$

Note that $X_1^n = 0$ and that the operators X_k^n, Γ_k^n are associated, respectively, to the interpolation and restriction. We can rewrite (4.4) to the standard form

$$e_k^n = G_k^n e_l^{n-1} + \psi_k^n, \qquad n=1,2,.... ; \quad k=1,....,l, \tag{4.6}$$

where the amplification operators G_k^n and the local perturbation terms ψ_k^n are defined by a recurrence relation:

$$\begin{aligned} G_1^n &= \Gamma_1^n, \\ G_k^n &= X_k^n\, G_{k-1}^n + \Gamma_k^n, \qquad k=2,....,l, \end{aligned} \tag{4.7}$$

$$\begin{aligned} \psi_1^n &= \phi_1^n, \\ \psi_k^n &= X_k^n\, \psi_{k-1}^n + \phi_k^n, \qquad k=2,....,l. \end{aligned} \tag{4.8}$$

The error recurrence (4.6) describes the error propagation for all refinement levels. The main interest lies in the operator G_l^n and the local perturbation ψ_l^n, since coarse grid values are always updated by fine grid values. In (4.6) this is reflected by the presence of e_l^{n-1}.

The stability of the implicit Euler method in the above is contained the following lemma:

LEMMA 4.1. Let ν be the logarithmic norm value defined in assumption (A4) of Section 2. Then,

$$\|(Z_1^n)^{-1}\| \le \frac{1}{1-\tau\nu}, \quad \forall \tau\nu < 1 \qquad k=1. \tag{4.9}$$

$$\|(Z_k^n)^{-1}\| \le \begin{cases} \dfrac{1}{1-\tau\nu} & , \ \forall \tau\nu < 1 \text{ if } \nu > 0, \\ 1 & , \ \forall\, \tau > 0 \text{ if } \nu \le 0, \end{cases} \qquad k=2,....,l.$$

PROOF. The result for $k=1$ is standard since D_1^n is the unit matrix (see [4], p.46). The premultiplication of M_k^n for $k>1$ with D_k^n has the effect that either entire rows of M_k^n are put to zero, or are left unchanged. From (2.3) we then can immediately deduce that for $\nu > 0$ the bound $(1-\tau\nu)^{-1}$ still holds, whereas for $\nu \le 0$ the zero rows introduce the bound 1. □

Observe that the replacement of the bound $(1-\tau\nu)^{-1}$ by the bound 1 for $\nu < 0$ implies that in this case we do no longer exploit the damping property of implicit

Euler. For the analysis to follow this is no restriction since we are here merely interested in proving stability and convergence results. Specifically the stability result we are going to prove is not dependent on the damping in implicit Euler. To shorten derivations, we first make another assumption.

(A5) The logarithmic norm bound ν from (A4) in is nonpositive. Hence we restrict ourselves to dissipative problems. This is not essential; results obtained for $\nu \leq 0$ can be extended to the case $\nu > 0$ by inserting $(1 - \tau\nu)^{-1}$ for the bound 1 any time the stability inequality $\|(Z_k^n)^{-1}\| \leq 1$ is used.

3.4.2. Stability and linear interpolation

In this section we will prove a general stability result for the multilevel adaptive grid method (3.3) that is similar to the stability result (2.8) for the Euler method applied without adaptation.

THEOREM 4.2. Let $\nu \leq 0$ according to (A5) and suppose that linear interpolation is used. Then, for all $\tau > 0$ and all $n \geq 1$,

$$\|G_k^n\| \leq 1, \quad k = 1, ..., l, \tag{4.10}$$

$$\|\psi_k^n\| \leq \sum_{j=1}^{k} \|r_j^n\|, \quad k = 1, ..., l, \tag{4.11}$$

$$\|e_l^n\| \leq \|e_l^{n-1}\| + \sum_{k=1}^{l} \|r_k^n\|. \tag{4.12}$$

PROOF. Inequality (4.12) is a trivial consequence of (4.10) and (4.11). Let us first prove (4.10). This is done by induction with respect to k. Suppose $\|G_{k-1}^n\| \leq 1$. From (4.7) it follows that

$$\|G_k^n\| = \|X_k^n G_{k-1}^n + \Gamma_k^n\| = \|(Z_{k-1}^n)^{-1} Q_k^n\| \leq \|Q_k^n\|, \tag{4.13}$$

where $Q_k^n = (I_k - D_k^n) P_{k-1k} G_{k-1}^n + D_k^n R_{lk}$.

Consider the i^{th} row of this operator. Suppose $(D_k^n)_{ii} = 1$. Then

$$\sum_j |(Q_k^n)_{ij}| = \sum_j |(R_{lk})_{ij}| = 1, \tag{4.14}$$

by definition of the restriction operator R_{lk}. Next suppose $(D_k^n)_{ii} = 0$. Then

$$\sum_j |(Q_k^n)_{ij}| = \sum_j |(P_{k-1k} G_{k-1}^n)_{ij}| \leq \|P_{k-1k} G_{k-1}^n\| \leq \|P_{k-1k}\| \|G_{k-1}^n\| \leq 1, \tag{4.15}$$

by virtue of the induction hypothesis and the norm

$$\|P_{k-1k}\| = 1 \tag{4.16}$$

of the linear interpolation operator P_{k-1k}. Combining (4.14) and (4.15) gives $\|Q_{k+1}^n\|$ and inequality (4.10) now follows from (4.13). The induction proof is finished if we can prove that $\|G_1^n\| \leq 1$. This follows immediately from the observation that $G_1^n = \Gamma_1^n = (Z_1^n)^{-1} R_{l1}$.

There remains to prove (4.11). We have $\|\phi_k^n\| \leq \|r_k^n\|$. It then follows from (4.8) that

$$\|\psi_k^n\| \leq \|X_k^n\| \|\psi_{k-1}^n\| + \|r_k^n\|, \tag{4.17}$$

so that we are finished if we can prove that $\|X_k^n\| \leq 1$. This is trivial due to (4.16) and $\|I_k - D_k^n\| = 1$. □

The inequality (4.12) is the counterpart of the inequality (2.8). We may conclude from Theorem 4.2 that when implicit Euler is stable and we interpolate linearly, our multilevel adaptive grid method (3.3) retains stability of implicit Euler through the bound $\|G_l^n\| \leq 1$.

3.4.3. Stability and higher-order interpolation

A drawback of linear interpolation is its limited accuracy. In a genuine application it might well be preferable to use higher-order interpolants (in [13] we have successfully used fourth-order Lagrangian interpolation). Unfortunately, in that case we must have $\|P_{k-1k}\| > 1$ so that we are not able to prove the results of Theorem 4.2 when following the above method of proof. If $\|P_{k-1k}\| > 1$, then it is possible to prove (a constrained form of) stability by introducing an additional condition that underlies the intention of interpolating exclusively in low-error regions. Unfortunately, this condition turns out to be of no direct practical use. On the other hand, numerical evidence suggests very strongly that those higher-order interpolants do not cause genuine stability problems in real application. We believe we owe this to the fact that the method interpolates in low-error regions, so that, loosely speaking, this condition is satisfied implicitly.

3.5. Error analysis

We will present a detailed examination of the local error. From this we deduce the refinement condition which henceforth underlies the refinement strategy. This condition enables us to control the contribution of spatial interpolation errors in favor of spatial discretization errors. Due to this condition, we can prove a convergence result as if we are working on a single, fixed grid. Specifically, it will be shown that the usual convergence behavior applies and that the accuracy obtained is comparable to the accuracy obtained on the finest grid if this grid would be used without any adaptation.

3.5.1. The local level error

Let $u_k(t)$ denote the pointwise restriction of the true solution $u(x,y,t)$ to ω_k. Consider (4.1). By replacing all $\tilde{U}$-values by associated u_k-values, the local perturbation r_k^n becomes the local level error at grid level k. For convenience, we will denote this error also by r_k^n:

$$r_k^n = u_k^n - D_k^n\,[R_{lk}u_l^{n-1} + \tau F_k(t_n,\,u_k^n)] - \\ (I_k - D_k^n)\,[P_{k-1k}u_{k-1}^n + b_k^n], \qquad n=1,2,\ldots.\,; \quad k=1,\ldots.,l, \tag{5.1}$$

where $u_k^n = u_k(t_n)$ and P_{01}, u_0^n, b_1^n are auxiliary and put to zero; r_k^n contains the following local error components, the local spatial error induced by the finite-difference approximation, the local temporal error of the implicit Euler method, and the interpolation error. We first discuss these different components. They are defined in the standard way by,

$$\alpha_k(t) = \frac{d}{dt}u_k(t) - F_k(t,u_k(t)) \quad \text{(spatial discretization error)} \tag{5.2}$$

$$\beta_k(t) = u_k(t) - u_k(t-\tau) - \tau\frac{d}{dt}u_k(t) \quad \text{(temporal error)} \tag{5.3}$$

$$\gamma_k(t) = u_k(t) - P_{k-1k}u_{k-1}(t) - b_k(t) \quad \text{(interpolation error)} \tag{5.4}$$

The grid function $b_k(t)$ in (5.4) has the same meaning as b_k^n in (3.5). In the following, we assume without loss of generality that $h_{x,k} = h_{y,k} = h_k$. In view of assumptions (A2) and (A3) made in Section 2, we have

$$\alpha_k^n(t) = O(h_k^2), \qquad h_k \to 0, \tag{5.5}$$

with order constants determined by higher-order spatial derivatives of u and by PDE operator quantities. Likewise, (A2) implies $\beta_k(t) = \tau^2 C_k$ where $C_k = -\frac{1}{2}\,d^2u_k/dt^2$ evaluated at a time $t+(\kappa-1)\tau$, $0 \le \kappa \le 1$. If u is a C^3-function in t, then

$$\beta_k(t) = -\frac{1}{2}\tau^2\frac{d^2}{dt^2}u_k(t) + O(\tau^3), \qquad \tau \to 0, \tag{5.6}$$

Let q denote the accuracy order of the (Lagrangian) interpolation. Then

$$\gamma_k(t) = O(h_k^q), \qquad k=2,\ldots.,l, \tag{5.7}$$

and here the order constants again depend exclusively on higher spatial derivatives of u, assuming sufficient differentiability. If linear interpolation is used, then assumption (A2) implies $q = 2$ and second-order spatial derivatives determine the constants.

Now, using the relation $u_k^{n-1} = R_{lk} u_l^{n-1}$ for $k=1,...,l$, we can derive

$$r_k^n = D_k^n(\tau\alpha_k^n + \beta_k^n) + (I_k - D_k^n)\gamma_k^n, \qquad n=1,2,.... ; \quad k=1,....,l. \tag{5.8}$$

Note, by definition of D_k^n, that $D_k^n(\tau\alpha_k^n + \beta_k^n)$ is the restriction of the usual local discretization error $\tau\alpha_k^n + \beta_k^n$ to the integration domain of the grid ω_k, while $(I_k - D_k^n)\gamma_k^n$ represents the restriction of the interpolation error γ_k^n to the complement of this domain.

3.5.2. A crude global error bound

Denote the global discretization error by $e_k^n = u_k^n - U_k^n$ and suppose $e_k^0 = 0$. For any choice of D_k^n the consistency results (5.5) - (5.8) imply

$$r_k^n = O(\tau h_k^2) + O(\tau^2) + O(h_k^q). \tag{5.9}$$

If we now suppose linear interpolation and assumption (A5), then application of (4.12) yields

$$\|e_l^n\| \le \|e_l^{n-1}\| + \|S^n\| \le \|S^1\| + ... + \|S^n\|, \qquad n=1,2,...., \tag{5.10}$$

where $\|S^n\| = O(\tau h_1^2) + O(\tau^2) + O(h_1^2)$. Here the coarsest mesh width occurs due to simply adding all normed local level errors in (4.15), including $\|r_1^n\|$. Following standard practice we thus obtain at any fixed time point $t_n = n\tau$ the global error bound

$$\|e_l^n\| \le C_1\tau + C_2 h^2 + C_3\frac{h^2}{\tau}, \tag{5.11}$$

where $h = h_1$, and C_1, C_2, C_3 are positive constants independent of step size and mesh sizes.

The first two terms are due to the temporal integration and spatial discretization. They will vanish if mesh sizes and step size tend to zero independently of each other, thus reflecting the unconditional convergence of the method when applied without adaptation. On the other hand, if no relation is imposed between τ and h, then the third term can grow unboundedly as τ, $h \to 0$. This term is due to the interpolation. Hence, even though we have stability and consistency, this result shows that unconditional convergence cannot be hoped for. Fortunately, this conclusion is not as bad as at it looks. By not specifying the matrices D_k^n and, subsequently, by adding norms of the local level errors, we have simply supposed arbitrary integration domains at all levels of refinement. This must lead to a crude error bound like (5.11). In application, the computations should be organized in such a way that the interpolation only takes place in low-error regions so that the interpolation error is virtually absent. This poses the task of setting up a precise error analysis and the design of a local refinement strategy aimed at a suitable selection of the matrices D_k^n.

3.5.3. Local and global errors

According to (4.6), the global error e_k^n satisfies the recurrence relation

$$e_k^n = G_k^n e_l^{n-1} + \psi_k^n, \qquad n=1,2,\ldots. ; \quad k=1,\ldots.,l, \tag{5.12}$$

where ψ_k^n is the local error defined by the recursion (cf. (4.8), (4.5))

$$\begin{aligned} \psi_1^n &= (Z_1^n)^{-1} r_1^n, \\ \psi_k^n &= X_k^n \psi_{k-1}^n + (Z_k^n)^{-1} r_k^n, \qquad k=2,\ldots.,l. \end{aligned} \tag{5.13}$$

The operators G_k^n, X_k^n and Z_k^n are supposed to be redefined (replace all $\tilde{U}$-values by associated u_k-values). Note that ψ_k^n is essentially different from the local level error r_k^n. While r_k^n is associated to the single k^{th} level, ψ_k^n is associated to all levels up to this k^{th} level according to (5.13). This recursion governs the propagation of each local level error when introducing higher-and-higher levels. Elaborating it gives, for $k=1,\ldots.,l$,

$$\psi_k^n = \sum_{j=1}^{k} \Big(\prod_{i=k}^{j+1} X_i^n\Big) (Z_j^n)^{-1} r_j^n. \tag{5.14}$$

Next we split ψ_k^n into its temporal and spatial part denoted by, respectively, $\psi_{k,t}^n$ and $\psi_{k,s}^n$:

$$\psi_k^n = \psi_{k,t}^n + \psi_{k,s}^n, \qquad k=1,\ldots.,l, \tag{5.15}$$

and it follows from (5.8) that $\psi_{k,t}^n$ and $\psi_{k,s}^n$ are given, respectively, by

$$\psi_{k,t}^n = \sum_{j=1}^{k} \Big(\prod_{i=k}^{j+1} X_i^n\Big) (Z_j^n)^{-1} D_j^n \beta_j^n \tag{5.16}$$

$$\psi_{k,s}^n = \sum_{j=1}^{k} \Big(\prod_{i=k}^{j+1} X_i^n\Big) (Z_j^n)^{-1} [\tau D_j^n \alpha_j^n + (I_j - D_j^n)\gamma_j^n] \tag{5.17}$$

Let us first examine $\psi_{k,t}^n$. Since β_k^n does not depend on mesh sizes, we have $\beta_k^n = R_{lk}\beta_l^n$. Substitution into (5.16) then yields

$$\psi_{k,t}^n = \sum_{j=1}^{k}(\prod_{i=k}^{j+1} X_i^n)\,(Z_j^n)^{-1} D_j^n R_{lj}\beta_l^n \tag{5.18}$$

and we see that this operator is just the amplification operator G_k^n featuring in (5.12); see the recursion (4.7). In conclusion, $\psi_{k,t}^n$ satisfies

$$\psi_{k,t}^n = G_k^n \beta_l^n, \qquad k=1,....,l. \tag{5.19}$$

We next examine $\psi_{k,s}^n$. Using the definition of X_k^n given in (4.5), we rewrite (5.17) as

$$\begin{aligned}\psi_{k,s}^n = (Z_k^n)^{-1}(I_k - D_k^n)P_{k-1k}\sum_{j=1}^{k-1}(\prod_{i=k-1}^{j+1} X_i^n)\ * \\ (Z_j^n)^{-1}[\tau D_j^n \alpha_j^n + (I_j - D_j^n)\gamma_j^n] + \\ (Z_k^n)^{-1}[\tau D_k^n \alpha_k^n + (I_k - D_k^n)\gamma_k^n] \\ = (Z_k^n)^{-1}[\tau D_k^n \alpha_k^n + (I_k - D_k^n)\rho_k^n], \qquad k=1,....,l,\end{aligned} \tag{5.20}$$

where

$$\rho_k^n = \gamma_k^n + P_{k-1k}\psi_{k-1,s}^n, \qquad k=2,....,l, \tag{5.21}$$

and $\rho_1^n = 0$. In (5.20) the spatial local discretization error $D_k^n\alpha_k^n$ committed on the integration domain of grid ω_k is separated from the spatial local error part $(I_k - D_k^n)\rho_k^n$ defined outside this domain. Hence, ρ_k^n collects all spatial error contributions defined on the grids ω_j $(1 \le j \le k-1)$, including discretization error α_j^n and interpolation error γ_j^n, together with γ_k^n on ω_k. This separation enables us to formulate a refinement condition which ensures that when a new grid level is introduced, the spatial local accuracy outside its integration domain will be smaller than or equal to the spatial accuracy on the integration domain itself. This distribution of local space errors is desirable, as we never return to grid points lying outside a current integration domain.

The refinement condition constraints the matrices D_k^n, and is taken to be

$$\|(Z_k^n)^{-1}\tau D_k^n \alpha_k^n\| \ge \frac{1}{c}\|(Z_k^n)^{-1}(I_k - D_k^n)\rho_k^n\|, \qquad n=1,2,....\,; \quad k=2,....,l, \tag{5.22}$$

where $c > 0$ is a parameter specified in Section 6. If (5.22) is true, then all errors $\psi^n_{k,s}$ satisfy

$$\|\psi^n_{k,s}\| \leq (1+c)\|(Z^n_k)^{-1}\tau D^n_k \alpha^n_k\| \tag{5.23}$$

and combining (5.12) with (5.15), (5.19) enables us to present the global error inequality

$$\|e^n_k\| \leq \|G^n_k\|\|e^{n-1}_l\| + \|G^n_k \beta^n_l\| + (1+c)\|(Z^n_k)^{-1}\tau D^n_k \alpha^n_k\|, \tag{5.24}$$
$$n=1,2,\ldots. ; \quad k=1,\ldots.,l.$$

The importance of the refinement condition (5.22) is reflected by the fact that in (5.24) the interpolation error contribution has been removed. This is in agreement with our goal of developing a local refinement strategy that generates refined subgrids such that the accuracy obtained on the final finest grid is comparable to the accuracy obtained if this finest grid would be used without adaptation. We will further elaborate on condition (5.22) in Section 6. Note that it suffices to consider (5.22) only for $k=l$, since it suffices to consider (5.23) and (5.24) for $k=l$.

3.5.4. Convergence and linear interpolation

Assuming linear interpolation and assumption (A5), as in Section 5.2, (5.24) can be rewritten as

$$\|e^n_k\| \leq \|e^{n-1}_l\| + \|\beta^n_k\| + (1+c)\tau\|\alpha^n_k\|, \quad n=1,2,\ldots. ; \quad k=1,\ldots.,l. \tag{5.25}$$

Hence, following the same derivation as carried out for (5.11), for the highest level l the global error bound

$$\|e^n_l\| \leq C_1\tau + C_2(1+c)h^2_l \tag{5.26}$$

results where C_1 and C_2 are positive constants independent of step size and mesh sizes. This bound is unconditional in the sense that it assumes no relation between step size and mesh sizes and, according to our goal, the smallest mesh width h_l occurs. We have recovered an error bound similar to the standard error bound for implicit Euler when applied on a single grid.

3.5.5. Convergence and higher-order interpolation

As pointed out in Section 4.3, for the case of higher-order Lagrangian interpolants a powerful stability result like that of Theorem 4.2 are not available. However assuming that higher-order interpolation in low-error regions does not severely damages stability, as is strongly supported by our practical experience, it is natural to impose the refinement condition (5.22) also in the case of higher-order interpolation. Note that in the derivation of (5.22) no a priori choice was made for the interpolants.

3.6. The refinement condition

3.6.1. Determining the integration domains

Condition (5.22) needs first to be elaborated into a workable form before it can be implemented for determining the integration domains. To begin with, we rewrite the error ρ_k^n as

$$\begin{aligned}\rho_k^n = \gamma_k^n + P_{k-1k}\sum_{j=1}^{k-1}\Big(\prod_{i=k-1}^{j+1} X_i^n\Big)\,(Z_j^n)^{-1}\tau D_j^n\alpha_j^n + \\ P_{k-1k}\sum_{j=2}^{k-1}\Big(\prod_{i=k-1}^{j+1} X_i^n\Big)\,(Z_j^n)^{-1}(I_j - D_j^n)\gamma_j^n, \qquad 2\le k\le l.\end{aligned} \tag{6.1}$$

Next, we rewrite the first sum as

$$\begin{aligned}P_{k-1k}\sum_{j=1}^{k-1}\Big(\prod_{i=k-1}^{j+1} X_i^n\Big)\,(Z_j^n)^{-1}\tau D_j^n\alpha_j^n = P_{k-1k}(Z_{k-1}^n)^{-1}\tau D_{k-1}^n\,\alpha_{k-1}^n + \\ P_{k-1k}\sum_{j=2}^{k-1}\Big(\prod_{i=k-1}^{j+1} X_i^n\Big)\,(Z_j^n)^{-1}(I_j - D_j^n)P_{j-1j}(Z_{j-1}^n)^{-1}\tau D_{j-1}^n\,\alpha_{j-1}^n,\end{aligned} \tag{6.2}$$

and substitute this expression into (6.1). It then follows that ρ_k^n can be written as

$$\rho_k^n = \lambda_k^n + P_{k-1k}\sum_{j=2}^{k-1}\Big(\prod_{i=k-1}^{j+1} X_i^n\Big)\,(Z_j^n)^{-1}(I_j - D_j^n)\lambda_j^n, \qquad k=2,....,l, \tag{6.3}$$

where

$$\lambda_j^n = \gamma_j^n + P_{j-1j}(Z_{j-1}^n)^{-1}\tau D_{j-1}^n\,\alpha_{j-1}^n, \qquad j=2,....,l. \tag{6.4}$$

The error function λ_j^n contains the interpolation error at level j and the prolongation of the spatial discretization error of level $j-1$ to level j. The derivation now rests upon monitoring the error $(Z_k^n)^{-1}(I_k - D_k^n)\rho_k^n$ occurring in (5.22) through monitoring all errors $(I_j - D_j^n)\lambda_j^n$, $j \le k$, occurring in (6.3). The idea is to select the matrices D_j^n such that the error functions $(I_j - D_j^n)\lambda_j^n$ become sufficiently small. This makes sense because if C_3, C_4 are stability constants such that

$$\|(Z_j^n)^{-1}\| \le C_3, \qquad \| \prod_{i=k-1}^{j+1} X_i^n \| \le C_4, \tag{6.5}$$

then

$$\begin{aligned} &\|(Z_k^n)^{-1}(I_k - D_k^n)\rho_k^n\| \le \\ &C_3(1 + \|P_{k-1k}\|(k-2)C_3C_4)\ \max_{2\le j\le k}\ \{\|(I_j - D_j^n)\lambda_j^n\|\}. \end{aligned} \tag{6.6}$$

Hence, if for $k=2,....,l$ the matrices D_k^n are selected such that

$$C_3(1 + \|P_{k-1k}\|(k-2)C_3C_4)\ \max_{2\le j\le k}\ \{\|(I_j - D_j^n)\lambda_j^n\|\} \le c\|(Z_k^n)^{-1}\tau D_k^n\alpha_k^n\|, \tag{6.7}$$

then the refinement condition (5.22) is satisfied.

In general, the stability constants C_3 and C_4 are unknown. However, if the dissipativity assumption (A5) is satisfied, then the constant $C_3 \le 1$. Furthermore, if we use linear interpolation, then (4.16) applies and also C_4 can be put equal to one, so that (6.7) simplifies to

$$\max_{2\le j\le k}\ \{\|(I_j - D_j^n)\lambda_j^n\|\} \le \frac{c}{k-1}\|(Z_k^n)^{-1}\tau D_k^n\alpha_k^n\|, \qquad k=2,....,l. \tag{6.8}$$

If assumption (A5) does not hold or higher-order interpolation is used, then C_3 and C_4 may be larger than one, but not with a considerable extent. C_3 shall in general be of moderate size in view of the excellent stability behavior of implicit Euler. Our practical experience with fourth-order Lagrangian interpolation is that higher-order interpolation is unlikely to yield instability problems, thus indicating that also $\|X_k^n\|$, and hence C_4, are of moderate size. That is why we proceed with (6.8) and also to use it in situations where (A5) may be violated and/or higher-order interpolation is used.

In application it suffices to impose (5.22) for $k=l$ only, so that (6.8) can be replaced by

$$\max_{2\le k\le l}\ \{\|(I_k - D_k^n)\lambda_k^n\|\} \le \frac{c}{l-1}\|(Z_l^n)^{-1}\tau D_l^n\alpha_l^n\|. \tag{6.9}$$

In order to satisfy this condition, estimates of λ_k^n have to be computed. Therefore, to create an extra safety margin, we replace (6.9) by the slightly more conservative condition

$$\|(I_k - D_k^n)\zeta_k^n\| \le \frac{c}{l-1}\|(Z_l^n)^{-1}\tau D_l^n \alpha_l^n\|, \qquad k=2,\dots,l, \tag{6.10}$$

where, per component, ζ_k^n is defined as

$$(\zeta_k^n)_i = |(\gamma_k^n)_i| + |(P_{k-1k}(Z_{k-1}^n)^{-1}\tau D_{k-1}^n \alpha_{k-1}^n)_i|. \tag{6.11}$$

Condition (6.10) will determine the integration domain of ω_k. Let Ω_k^n be this integration domain and recall that when a node belongs to Ω_k^n, the corresponding diagonal entry of D_k^n is equal to one and zero otherwise. Suppose that the maximal level number l and $c\,(l-1)^{-1}\|(Z_l^n)^{-1}\tau D_l^n\alpha_l^n\|$ are known and that a solution at Ω_{k-1}^n, $k \le l$, has just been computed. Prior to the integration step on level k, our task is then to determine Ω_k^n. That is, we must define D_k^n such that (6.10) is satisfied and in such a way that the area of Ω_k^n is as small as possible. The actual selection of Ω_k^n is carried out by a flagging procedure which scans level-k grid points. A point is flagged if, using appropriate estimates,

$$(\zeta_k^n)_i > \frac{c}{l-1}\|(Z_l^n)^{-1}\tau D_l^n \alpha_l^n\|. \tag{6.12}$$

Hence, for such a point the corresponding diagonal entry $(D_k^n)_{ii} = 1$ and for non-flagged points we define $(D_k^n)_{ii} = 0$. This way the refinement condition (6.10) is satisfied.

In conclusion the solution at a node of grid ω_k is interpolated only if a corresponding component of ζ_k^n is smaller than the maximum of the spatial discretization error at the finest grid multiplied with $\tau c\,(l-1)^{-1}$. Otherwise integration is carried out at this node. No doubt this imposes a severe restriction on the size of the interpolation errors. On the other hand, this restriction is natural because when going to a higher level within the current time step, we never return to a grid point where the solution has been interpolated which means that the interpolation error will be carried along to the next time step. The fact that we do not return is a direct consequence of the nesting property of the integration domains, which we discuss next.

3.6.2. Restricted interpolation and the nesting property

We now introduce the nesting property of the integration domains. Recall that this property, being hidden in the definition of the matrices D_k^n, has played no role in the foregoing analysis. We stipulate that in application the nesting is enforced by the flagging procedure, in other words, this procedure scans only level-k points lying within the previous integration domain Ω^n_{k-1}. A direct consequence is that, different from (3.5), the interpolation is carried out only for level-k points within Ω^n_{k-1}. Here we will justify the deviation due to this restricted interpolation. We argue that the restricted interpolation is in fact allowed by the inequality (6.10) where interpolation over the whole of ω_k is still assumed.

Consider the error $(Z^n_{k-1})^{-1}\tau D^n_{k-1}\alpha^n_{k-1}$. This spatial error is defined at level $k-1$ and, by definition of D^n_{k-1}, has zero components outside Ω^n_{k-1}. Hence, all its prolongated components are taken into account in the flagging procedure for determining Ω^n_k. For the interpolation error γ^n_k,which lives on the whole of ω_k (grid expansion), the situation is different. However, restricted interpolation is allowed if for all level-k points outside Ω^n_{k-1}, the interpolation error satisfies

$$|(\gamma^n_k)_i| \leq \frac{c}{l-1}\|(Z^n_l)^{-1}\tau D^n_l\alpha^n_l\|, \tag{6.13}$$

because then points outside Ω^n_{k-1} will not be flagged if the interpolation step (3.5) would be carried out on the whole of ω_k. In other words, if (6.13) holds outside Ω^n_{k-1}, then the integration domains found with the restricted interpolation over Ω^n_{k-1} are equal to the domains found if the interpolation would be carried out on the whole of ω_k, which is in accordance with the method description (3.3).

The following argument shows that inequality (6.13) is very plausible with the restricted interpolation procedure. First we recall that Ω^n_1 coincides with the entire physical domain. Hence for k=2 there is no restricted interpolation so that for all level-2 points outside Ω^n_2 inequality (6.13) is trivially satisfied. Next consider the case k=3. Now the interpolation is restricted to level-3 points within Ω^n_2. Since for all level-2 points outside Ω^n_2 inequality (6.13) is satisfied, we are justified in supposing that this is also true for all level-3 points outside Ω^n_2, in view of the consistency of the interpolation (level-3 interpolation errors are smaller than level-2 errors). Further, by construction of Ω^n_3, (6.10) is satisfied for all level-3 points within Ω^n_2 and outside Ω^n_3, and so is (6.16). In conclusion, we may suppose that (6.13) is satisfied for all level-3 points outside Ω^n_3 when using the restricted interpolation for k=3. For k=4 and so on this argument can be repeated.

3.6.3. Implementation aspects

On top of the flagging procedure implementing (6.12) some safety measures have been built. Any node for which (6.12) is true is flagged together with its eight neighbors. Next, to create an extra buffer, all sides of cells with at least one flagged corner node are bisected. This means that a buffer zone of two mesh widths is used around any intolerable node. Near boundaries, physical and internal ones, the buffering differs slightly. Although in theory this buffering could be omitted, in practice it is wise to create a buffer zone around intolerable nodes because the estimation of higher spatial derivatives contained in α^n_k and γ^n_k is prone to inaccuracies. After the flagging procedure, a cluster algorithm groups all untolerable nodes together to form the newly defined integration domain.

The parameter c in (6.12) must be specified. In view of result (5.25), c should be taken small so that the spatial accuracy obtained is indeed nearly equal to the spatial accuracy obtained without adaptation. In fact hand, the smaller c, the more points will be flagged and hence the safer the local refinement will be (c=0 implies global refinement). On the other hand, when c is too large, the situation can occur that the space errors are large and refinement is neccessary but no nodes are flagged because (6.12) is satisfied at every node. Hence, c is available as a tuning parameter. In the experiments in Section 7 we have simply put c=1.

Estimates of spatial interpolation and discretization errors are required. For $2 \le k \le l$ we must estimate the interpolation error γ_k^n and the prolongated spatial discretization error $P_{k-1k}(Z_{k-1}^n)^{-1} D_{k-1}^n \alpha_{k-1}^n$. Further, an estimate of the spatial discretization error $(Z_l^n)^{-1} D_l^n \alpha_l^n$ committed at the final l^{th} level must be available at all lower levels. Because we use local, uniform grids, the estimation of these errors can be realized cheaply and easily. Consider the error α_k^n (cf. (5.2)) and let p be the order of consistency (in this paper p=2). The estimation we apply is based on the use of a second spatial discretization operator $\tilde{F}$ of a higher-order $\tilde{p}$. After some elementary calculations we obtain the following approximation

$$\alpha_k^n \approx \tilde{F}_k(t_n, u_k(t_n)) - F_k(t_n, u_k(t_n)) \tag{6.14}$$

as an asymptotically correct estimator for α_k^n. The benefit of using uniform grids now lies in the fact that $\tilde{F}$ is easily constructed. At internal nodes our $\tilde{F}$ provides fourth-order accuracy (standard symmetrical differences), while at nodes adjacent to physical or internal boundaries third-order accuracy is realized (standard one-sided differences). The benefit of using uniform grids is also reflected in the estimation for the error γ_k^n (cf. (5.4)). So far we have implemented Lagrangian interpolation of second (linear) and fourth order. For the second-order interpolation we need to estimate spatial derivatives u_{xx}, etc., while in the fourth-order case spatial derivatives like u_{xxxx} appear. For both cases the estimation is straightforward.

We emphasize that, in spite of its simplicity, linear interpolation may become disadvantageous due to the low order of accuracy. Inspection of the various terms in (6.12) suggests to compare the following order relations:

$$\tau(P_{k-1k}(Z_{k-1}^n)^{-1} D_{k-1}^n \alpha_{k-1}^n)_i = O(\tau h_k^2), \tag{6.15a}$$

$$\|(Z_l^n)^{-1} \tau D_l^n \alpha_l^n\| = O(\tau h_l^2), \tag{6.15b}$$

$$(\gamma_k^n)_i = O(h_k^2), \qquad \text{second-order linear,} \tag{6.15c}$$

$$(\gamma_k^n)_i = O(h_k^4), \qquad \text{fourth-order Lagrangian.} \tag{6.15d}$$

In the discretization terms the step size τ is contained. Consequently, it is the interpolation error that may will govern the refinement if τ is very small, and particularly so when the interpolation is linear. The comparison is clearly in favor of the fourth-order interpolation.

To estimate the right-hand-side term $\|(Z_l^n)^{-1} D_l^n \alpha_l^n\|$ of (6.12) for $2 \leq k \leq l-1$, we exploit the asymptotics. Since the mesh width of level k is half that of level $k+1$, we thus invoke

$$\|(Z_l^n)^{-1} D_l^n \alpha_l^n\| \approx 2^{-p\,(l-k)} \|(Z_k^n)^{-1} D_k^n \alpha_k^n\|, \qquad l \geq k+1, \tag{6.16}$$

for $k=1,2,\ldots$. In theory it suffices to do this only for $k=1$, but since for larger values of k this estimation will become more and more accurate, it is done for every k.

Finally, we make a few remarks about the approximations (6.14) and (6.16). Our method, like every other adaptive grid method is designed to solve PDEs with steep solutions. Yet (6.14) and (6.16) underlie asymptotics, which means that they are only accurate if the solution is sufficiently smooth on the grid in use. This constitutes a problem for LUGR methods, because these methods estimate errors on coarse grids. Nevertheless, if in practice the estimated error is not that accurate, it might still give an good indication of where the spatial error is large and where it is not and, specifically, the estimated error might still be in more or less the same order of magnitude as the exact error, in which case the implemented refinement strategy based upon (6.14) will still work. In our experience so far this is indeed the case. We believe this is due to the fact that estimation (6.16) is carried out for finer-and-finer subgrids with an increasing accuracy which partly remedies the problem. However, if solutions become very steep it may be necessary to improve the implementation of the refinement strategy.

3.7. Numerical example

This section is devoted to an illustration of the foregoing error analysis. Our goal here is to numerically illustrate that by imposing the refinement condition the usual order behavior is recovered. At the same time, the spatial accuracy obtained is comparable to the spatial accuracy on the finest grid if this grid would be used without adaptation.

3.7.1. The issue of implicitness

We use the implicit Euler method for time integration. In connection with implicitness, two points are worth mentioning. The first is that at any time step refinement takes place at different levels, resulting in a different Jacobian per level whose order usually varies. This impedes the profitable use of old Jacobians (like in sophisticated stiff ODE solvers), unless it is decided not to adapt grids at every time step, but instead per prescribed number of steps. We consider this as part of an overall strategy that can easily be placed on top of the existing one. We adapt grids at every time step since our main aim with the experiments is to illustrate the convergence analysis together with the refinement strategy. However, when dealing with real applications, it is most likely to be more advantageous to omit adaptation at every base time step, just for efficiency reasons. The second point is that the Jacobians do not posses a regular band structure, since the integration domains Ω_k^n normally have an irregular shape. Unlike the first, this point is intrinsic to the local refinement method. In the experiments reported here the Harwell sparse matrix solver MA28 has been used. This solver is well suited to cope with the structures we meet, but is rather time consuming for the present application. It is likely that standard iterative methods can be applied at lower costs.

3.7.2. The example problem

This test example is hypothetical and due to [1]. The equation is the linear parabolic equation

$$u_t = u_{xx} + u_{yy} + f(x,y,t), \qquad 0<x,y<1, \qquad t>0, \tag{7.1}$$

and the initial function, the Dirichlet boundary conditions for and the source term f are selected so that the exact solution is

$$u(x,y,t) = \exp[-80((x - r(t))^2 + (y - s(t))^2)], \tag{7.2}$$

where $r(t) = \frac{1}{4}[2 + \sin(\pi t)]$ and $s(t) = \frac{1}{4}[2 + \cos(\pi t)]$. This solution is a cone that is initially centered at $(\frac{1}{2},\frac{3}{4})$ and that symmetrically rotates around $(\frac{1}{2},\frac{1}{2})$ in a clockwise direction with a constant speed. We have used this problem to subdue our refinement method to a convergence test. Observe that the semi-discrete version of this problem satisfies the dissipativity assumption (A5).

3.7.3. Convergence experiments

We have carried out two identical convergence experiments. In the first linear interpolation has been used and in the second, fourth-order Lagrangian. In both the solution is computed four times over the interval $0 \le t \le 2$, using a uniform 10×10 grid and a constant time step size τ. In the first computation $l=1$, in the second $l=2$, and so on. Since per computation the smallest mesh width is halved, τ is simultaneously decreased by 2^2 in view of the first order of implicit Euler. Hence, in line with our analysis, per computation the maximal global error should also decrease by 2^2.

Table 7.1 shows the maxima of the global errors restricted to the finest integration domain in use. This table clearly reveals the expected order behavior. The errors of the $l=4$ runs are about a factor 4 smaller than the corresponding errors of the $l=3$ runs. Note that there is hardly a difference between the corresponding errors, showing that, as anticipated by our strategy, the choice of interpolant has no notable influence on the error. We emphasize that, in spite of the relatively large values for τ, the spatial error dominates the global errors shown in this table. For example, using $\tau = 0.125$ instead of $\tau = 0.5$ in the $l=2$ run, the same global errors are found

τ	# of levels	interpolation	single grid	t			
				0.50	1.00	1.50	2.00
2.00000	1		10×10				0.16447
0.50000	2	linear		0.03876	0.03890	0.03891	0.03891
		fourth order		0.03929	0.03945	0.03946	0.03946
			20×20	0.03865	0.03881	0.03882	0.03882
0.12500	3	linear		0.01369	0.01369	0.01369	0.01369
		fourth order		0.01376	0.01376	0.01376	0.01376
			40×40	0.01389	0.01389	0.01389	0.01389
0.03125	4	linear		0.00340	0.00340	0.00340	0.00340
		fourth order		0.00359	0.00359	0.00359	0.00359
			80×80	0.00347	0.00347	0.00347	0.00347

TABLE 7.1. Maxima of global errors restricted to the finest domain. Comparison with errors on a standard uniform grid.

(they deviate in the third or fourth decimal digit). In other words, conclusions on the spatial error behavior induced by the local refinement algorithm can be drawn from this table.

These results convincingly show, for the current example problem, that the use of the refinement condition ensures that the spatial accuracy obtained is very much comparable to the spatial accuracy on the finest grid if this grid is used without any adaptation. Finally we note that the choice $c=1$ apparently has no influence on the error. We owe this to the fact that the refinement condition has been derived from errors bounds and is thus rather conservative.

The use of the two different interpolants is expressed in the slightly different integration domains shown in Figures 7.1 and 7.2. As expected, at the higher levels linear interpolation gives rise to somewhat larger domains. Showing that linear interpolation is more expensive. As a rule, fourth-order interpolation is to be preferred as it leads to smaller domains. Note that for both interpolants the moving domains accurately reflect the symmetric rotation of the cone, which once again nicely illustrates the reliability of the implemented refinement condition with the various estimators.

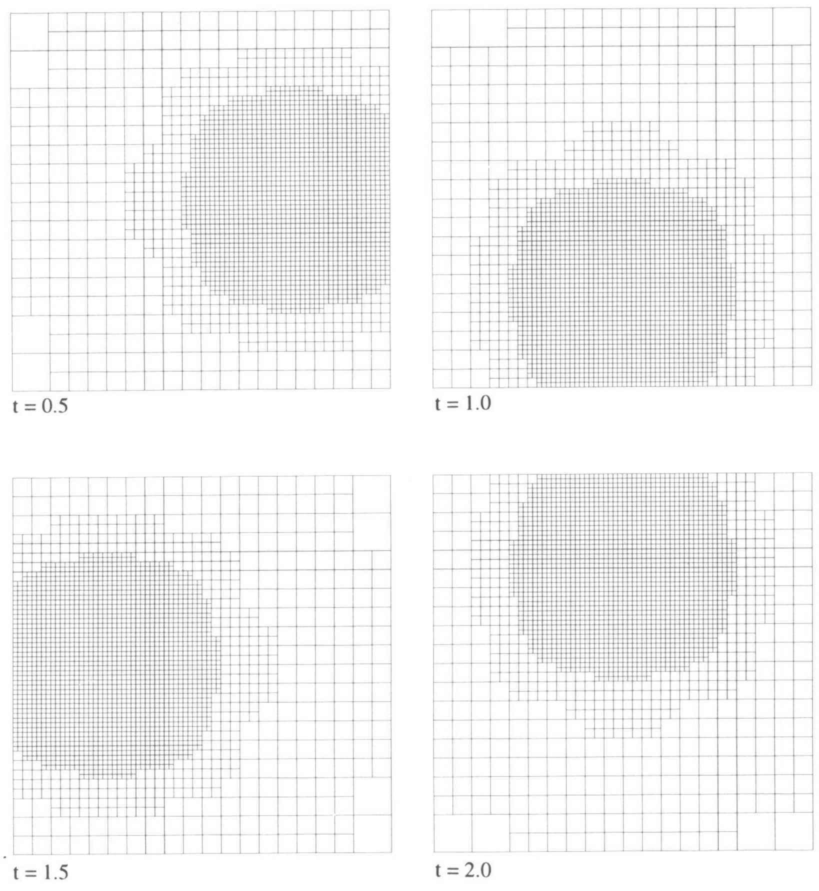


FIGURE 7.1. Linear interpolation. Integration domains for the l=4 run at four different times. The size of the integration domains decreases only slowly with the number of levels. This is due to the fact that the cone is not very steep. At the end time t=2.0 the number of nodes amounts to 121, 425, 813 and 1917, respectively.

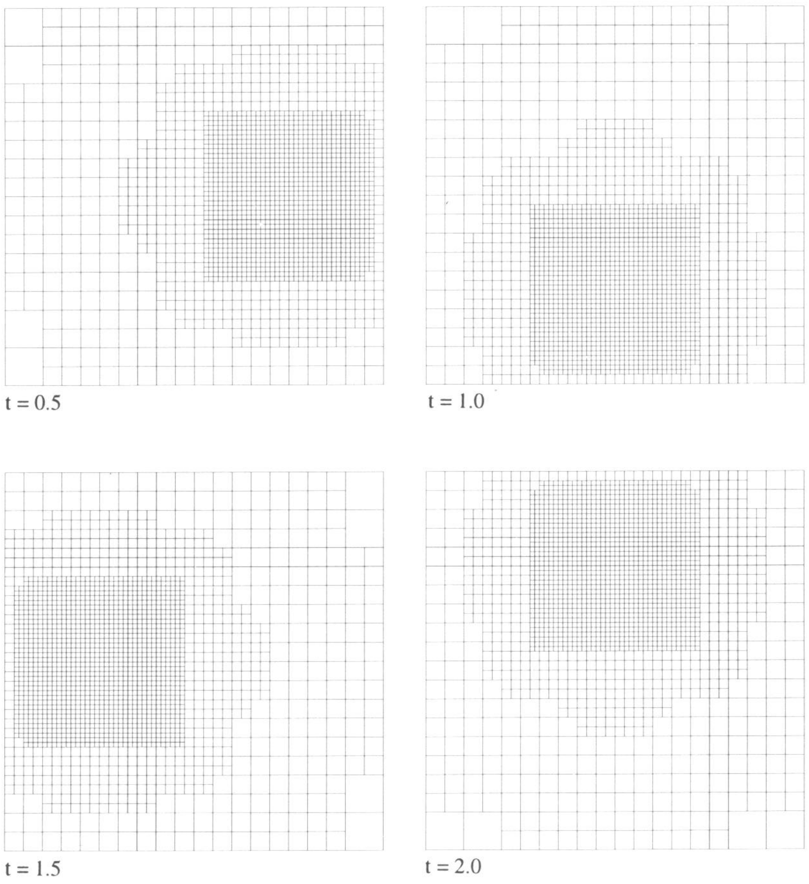


FIGURE 7.2. Fourth-order interpolation. Integration domains for the l=4 run at four different times. At the end time t=2.0 the number of nodes amounts to 121, 425, 813 and 1361, respectively.

3.8. Final remarks and future plans

In our future research we plan to pay more attention to time-stepping efficiency. Using the refinement strategy of this paper as a starting point, we plan to examine the application of methods possessing a higher order in time. Natural candidates belong to the class of Runge-Kutta methods. It should be stressed, though, that fully implicit methods can only be of serious advantage if the numerical algebra issue can be satisfactorily solved. In this connection splitting methods of the ADI and LOD type (see [9]) may therefore provide an attractive alternative to fully implicit ones, although they are usually less accurate in time. Another point of serious practical concern is to apply methods not only using an a priori chosen number of levels, but to have also the possibility to vary the number of levels. This might be useful for the computation of solutions which, for example, steepen up in time like the combustion problem in [13]. For such problems, the application of a variable number of levels should be combined with the use of variable temporal step sizes. Preferably, the complete adaptation then should be monitored by estimators of temporal and spatial errors in such a way that there is a balance between the two which aims at minimizing the waste of computing time.

Acknowledgement
We would like to thank Willem Hundsdorfer for his careful reading of the manuscript.

References

1. S. Adjerid and J.E. Flaherty (1988). A local Refinement Finite Element Method for Two Dimensional Parabolic Systems, *SIAM J. Sci. Statist. Comput.*, 9, 792-811.
2. D.C. Arney and J.E. Flaherty (1989). An Adaptive Local Mesh Refinement Method for Time-Dependent Partial Differential Equations, *Appl. Numer. Math.*, 5, 257-274.
3. M.J. Berger and J. Oliger (1984). Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, *J. Comput. Phys.*, 53, 484-512.
4. K. Dekker and J.G. Verwer (1984). *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, North-Holland, Amsterdam-New York-Oxford.
5. R.E. Ewing (1989). Adaptive Grid Refinement for Transient Flow Problems, in *Adaptive Methods for Partial Differential Equations*, 194-205, ed. J.E. Flaherty, P.J. Paslow, M.S. Shephard, J.D. Vasilakis, SIAM Publications, Philadelphia.
6. W.D. Gropp (1980). A Test of Moving Mesh Refinement for 2D-Scalar Hyperbolic Problems, *SIAM J. Sci. Statist. Comput.*, 1, 191-197.
7. W.D. Gropp (1987). Local Uniform Mesh Refinement with Moving Grids, *SIAM J. Sci. Statist. Comput.*, 8, 292-304.

8. W.D. GROPP (1987). Local Uniform Mesh Refinement on Vector and Parallel Processors, in *Large Scale Scientific Computing*, 349-367, ed. P. DEUFLHARD, B. ENGQUIST, Birkhäuser Series Progress in Scientific Computing.
9. W.H. HUNDSDORFER and J.G. VERWER (1989). Stability and Convergence of the Peaceman-Rachford ADI Method, *Math. Comp.*, 53, 81-101.
10. S. MCCORMICK and J.W. THOMAS (1986). The Fast Adaptive Composite Grid (FAC) Method for Elliptic Equations, *Math. Comp.*, 46, 439-456.
11. J.E. FLAHERTY, P.K. MOORE and C. OZTURAN (1989). Adaptive Overlapping Grid Methods for Parabolic Systems, in *Adaptive Methods for Partial Differential Equations*, 176-193, ed. J.E. FLAHERTY, P.J. PASLOW, M.S. SHEPHARD, J.D. VASILAKIS, SIAM Publications, Philadelphia.
12. J.M SANZ-SERNA and J.G. VERWER (1989). Stability and Convergence at the PDE/Stiff ODE Interface, *Appl. Numer. Math.*, 5, 117-132.
13. R.A. TROMPERT and J.G. VERWER (1991). A Static-Regridding Method for Two Dimensional Parabolic Partial Differential Equations, *Appl. Numer. Math.*, 8, 65-90.

Chapter 4

Runge-Kutta Methods and Local Uniform Grid Refinement

R.A. Trompert and J.G. Verwer
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

Local uniform grid refinement (LUGR) is an adaptive grid technique for computing solutions of partial differential equations possessing sharp spatial transitions. Using nested, finer-and-finer, uniform subgrids, the LUGR technique refines the space grid locally around these transitions, so as to avoid discretization on a very fine grid covering the entire physical domain. This paper examines the LUGR technique for time-dependent problems when combined with static regridding. Static regridding means that in the course of the time evolution, the space grid is adapted at discrete times. The present paper considers the general class of Runge-Kutta methods for the numerical time integration. Following the method-of-lines approach, we develop a mathematical framework for the general Runge-Kutta LUGR method applied to multi-space dimensional problems. We hereby focus on parabolic problems, but a considerable part of the examination applies to hyperbolic problems as well. Much attention is paid to the local error analysis. The central issue here is a 'refinement condition' which is to underly the refinement strategy. By obeying this condition, spatial interpolation errors are controlled in a manner that the spatial accuracy obtained is comparable to the spatial accuracy on the finest grid if this grid would be used without any adaptation. A diagonally-implicit Runge-Kutta method is discussed for illustration purposes, both theoretically and numerically.

published in: Math. Comp. **60** (202) (1993), 591-616

4.1. Introduction

Local uniform grid refinement (LUGR) is an adaptive grid technique for computing solutions of partial differential equations (PDEs) possessing sharp spatial transitions. Using nested, finer-and-finer, uniform subgrids, the LUGR technique refines the space grid locally around these transitions to avoid discretization on a very fine grid covering the entire domain. In this paper we examine the LUGR technique for time-dependent problems. Thus, typical solutions aimed at are those possessing sharp moving transitions, such as steep fronts, emerging layers, moving pulses, etc. For time-dependent problems, LUGR is combined with static regridding. Static regridding means that in the course of the time evolution, the space grid is adapted at discrete times.

We consider Runge-Kutta methods for the time integration and, following the method-of-lines approach, develop a mathematical framework for the general Runge-Kutta LUGR method. We hereby focus on parabolic problems, but a considerable part of the discussion applies to hyperbolic problems as well. The present paper is a continuation of [12] which deals with the implicit Euler method. Here we discuss how the ideas developed in [12] are extended to the general Runge-Kutta case. Like in [12], much attention is paid to the local error analysis. The central issue here is the 'refinement condition', which is to underly the refinement strategy. By obeying this condition, spatial interpolation errors are controlled in a manner that the spatial accuracy obtained is comparable to the spatial accuracy on the finest grid if this grid would be used without any adaptation. Non-numerical subjects, such as the data structure and the memory use, are not discussed here. These are the same as in [11]. For related earlier work on LUGR methods, we refer to *Berger* and *Oliger* [3], *Gropp* [6, 7], *Arney* and *Flaherty* [2] and references therein.

Section 2 is devoted to the method formulation. Here we develop the mathematical framework that enables us to give a concise description of the Runge-Kutta LUGR method. In Section 3 we set up a general error scheme, which is further elaborated in Sections 4 and 5. Section 4 briefly addresses the stability issue, while Section 5 is devoted to the local error analysis. Here we derive the important refinement condition. Under a natural assumption on the Runge-Kutta method, we next prove that the 'uniform in h' temporal order of the method is at least equal to the stage order. Noteworthy is that Sections 3 - 5 apply to the whole class of Runge-Kutta methods. As a result, the outcome of the analysis is of a general nature, so that for a specific Runge-Kutta method a further elaboration is needed. Such an elaboration is presented in the remainder of the paper for a three-stage diagonally-implicit Runge-Kutta (DIRK) method. In Section 6 attention is given to the order reduction phenomenon and to how to implement the refinement condition for this specific method. Section 7 deals with two numerical examples in two space dimensions. Finally, we conclude the paper with Section 8 discussing two important matters of practical interest.

4.2. THE GENERAL METHOD FORMULATION

4.2.1. The Runge-Kutta method

Consider the initial value problem for a standard system ordinary differential equations (ODEs)

$$\frac{d}{dt}U(t) = F(t, U(t)), \qquad 0 < t \leq T, \qquad U(0) = U^0. \tag{2.1}$$

The general one-step, s-stage RK scheme for the numerical solution of (2.1) is denoted by

$$U^{(i)} = U^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F(t_{n-1} + c_j \tau,\, U^{(j)}), \qquad 1 \le i \le s, \tag{2.2}$$

$$U^{n} = U^{n-1} + \tau \sum_{i=1}^{s} b_{i} F(t_{n-1} + c_i \tau,\, U^{(i)}), \tag{2.3}$$

where the step size τ may vary with n. Superscripts will refer to time, while bracketed superscripts are used for approximations at intermediate stages. As usual, we suppose $c_i = a_{i1} + \cdots + a_{is}$. In the remainder it is convenient to combine (2.2)-(2.3) into one formula. Denote $a_{s+1i} = b_i$, $1 \le i \le s$, $U^{(s+1)} = U^n$, then we rewrite (2.2)-(2.3) as

$$U^{(i)} = U^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F(t_{n-1} + c_j \tau,\, U^{(j)}), \qquad 1 \le i \le s+1. \tag{2.4}$$

4.2.2. The semi-discrete problem

Consider an initial-boundary value problem in d space dimensions,

$$u_t = L(t,u), \qquad 0 < t \le T, \qquad u(\underline{x},0) = u_0(\underline{x}), \tag{2.5}$$

where L is supposed to be of at most second order and provided with appropriate boundary conditions on the boundary $\partial\Omega$ of the space domain Ω. The boundary is taken to be locally parallel to the co-ordinate axes. The function $u(\underline{x},t)$ may be vector valued and is supposed to exist uniquely and to be as often differentiable on $(\Omega \cup \partial\Omega) \times [0,T]$ as the numerical analysis requires.

LUGR methods use local, uniform grids whose size and number mostly vary in time. Therefore, LUGR methods generate a sequence of operations on vectors in vector spaces with a variable dimension. This complicates the error analysis. In [12] we got round this problem by expanding the fine grids in the mathematical formulation of the method, so that the entire domain is covered. Also here we use this 'grid expansion'. Temporal integration then takes place on one part of the expanded fine grid and interpolation on the other. Note that this grid expansion does not take place in the actual application but only in the mathematical formulation of the method. Nevertheless, the results of the error analysis presented remain valid for the applicated method.

Let $l \in \mathbb{N}^+$. For $k = 1, \cdots, l$ we introduce uniform space grids ω_k where each ω_k is supposed to cover the whole of the interior domain Ω. The grid ω_k has no points on $\partial\Omega$. The grid ω_1 is called the base grid and, given this grid, ω_2 is obtained from ω_1 by bisecting all sides of all cells of ω_1, etc.. To (2.5) we now associate on each ω_k a real Cauchy problem for an explicit ODE system in $\mathbb{R}^{d_k}$,

$$\frac{d}{dt}U_k(t) = F_k(t, U_k(t)), \qquad 0 < t \le T, \qquad U_k(0) = U_k^0, \tag{2.6}$$

defined by a finite-difference space-discretization of (2.5) and its boundary conditions. Thus, U_k and F_k are vectors representing the values of grid functions defined on the grid ω_k. Each component of U_k and F_k itself is vector valued if u is vector valued. The boundary conditions have been worked into the semi-discrete system by eliminating semi-discrete values at $\partial\Omega$. The dimension d_k is determined by the spatial dimension, the grid spacing, and the number of PDEs. The initial vector U_k^0 for (2.6) is supposed to be exact.

In the remainder we let S_k with dim $(S_k) = d_k$ denote the grid function space. S_k coincides with $\mathbb{R}^{d_k}$ and U_k, F_k are elements of S_k. Let $u_k(t) \in S_k$ represent the natural (nodal wise) restriction of $u(x,t)$ to ω_k. In S_k the fully continuous problem (2.5) and the semi-discrete problem (2.6) are related by the local spatial discretization error

$$\alpha_k(t) = \frac{d}{dt}u_k(t) - F_k(t, u_k(t)), \qquad 0 \le t \le T. \tag{2.7}$$

In particular, u_k and α_k are sufficiently often differentiable with respect to t and $\alpha_k(t)$ has the order of consistency of the finite-difference scheme. Finally, we note once more that we consider elements $u_k(t)$, $U_k(t) \in S_k$ defined on space grids ω_k which cover the entire physical domain Ω.

4.2.3. The multi-level multi-stage RK method

Starting at the coarse base grid ω_1, this method successively integrates on subgrids of ω_k for $k = 2, \cdots, l$ over the same time interval $[t_{n-1}, t_n]$. Characteristic for the method is that subgrids, henceforth called the integration domains, are nested and that so to say on each domain a new initial-boundary value problem is solved. Required initial values are defined by interpolation from the next coarser integration domain or taken from a possibly existing one from the previous time interval. Boundary values required at internal boundaries are also interpolated from the next coarser integration domain. At each level of refinement, the domains are allowed to be disjunct and thus may consist of two or more subdomains. The nesting is continued up to a level fine enough to resolve the anticipated fine scale structure. This means that, given ω_1, l must be chosen sufficiently large. Having completed the integration on the finest, l^{th}-level integration domain, the process is repeated for the next time interval $[t_n, t_{n+1}]$ by again starting from ω_1. We note that all refined subgrids computed at forward time are kept in storage as they are used for step continuation. Further, for step continuation always the most accurate solution is used that is available.

The above described process is defined by the formulas

$$U_1^{(i)} = R_{l1} U_l^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F_1(t_{n-1} + c_j \tau, U_1^{(j)}), \tag{2.8a}$$

$$1 \le i \le s+1\ ; \quad k = 1,$$

$$U_k^{(i)} = D_k^n \, [R_{lk} U_l^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F_k(t_{n-1} + c_j \tau, U_k^{(j)})] + \tag{2.8b}$$

$$(I_k - D_k^n) \, [P_{k-1k} U_{k-1}^{(i)} + b_k^{(i)}], \quad 1 \le i \le s+1\ ; \ 2 \le k \le l,$$

where $U_k^{(s+1)} = U_k^n \in S_k$ is the approximation to $u_k(t_n)$ at the grid ω_k, $U_k^{(i)} \in S_k$ is the i^{th} intermediate approximation at ω_k, I_k: $S_k \to S_k$ is the unit matrix, D_k^n: $S_k \to S_k$ is a diagonal matrix with entries $(D_k^n)_{ii}$ either unity or zero, R_{lk}: $S_l \to S_k$, $k = 1, \cdots, l$, is the natural restriction operator from ω_l to ω_k with $R_{ll} = I_l$, P_{k-1k}: $S_{k-1} \to S_k$, $k = 2, \cdots, l$, is an interpolation operator from ω_{k-1} to ω_k, and $b_k^{(i)} \in S_k$ contains time-dependent terms emanating from the physical boundary $\partial\Omega$.

The nesting property of the integration domains is induced by the grid strategy. This strategy determines at which nodes integration or interpolation is carried out and defines the diagonal matrices D_k^n. If at a node integration is to take place, then the associated diagonal entry $(D_k^n)_{ii}$ is defined as $(D_k^n)_{ii} = 1$. For all remaining interpolation nodes, $(D_k^n)_{ii} = 0$. The nesting property itself cannot be recovered from the above formulation, as this is hidden in the actual definition of D_k^n.

The interpolation step on level $k \ge 2$ stands on its own and is represented by

$$(I_k - D_k^n) U_k^{(i)} = (I_k - D_k^n) \, [P_{k-1k} U_{k-1}^{(i)} + b_k^{(i)}], \qquad 1 \le i \le s+1. \tag{2.9}$$

The grid function $b_k^{(i)}$ plays an auxiliary role. We need to include it as boundary conditions have been worked into (2.6) (method of lines). For the analysis presented $b_k^{(i)}$ plays no role (it contains merely time-dependent terms and does not depend on $u(\underline{x},t)$). Likewise, the integration step on the integration domain of level k is represented by

$$D_k^n U_k^{(i)} = D_k^n \, [R_{lk} U_l^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F_k(t_{n-1} + c_j \tau, U_k^{(j)})], \tag{2.10}$$

$$1 \le i \le s+1,$$

where, according to (2.8a), $D_1^n = I_1$. Values at or beyond internal boundaries needed in the function evaluation in (2.10) are defined by (2.9), for each RK stage. Hence, due to the internal boundaries, (2.10) cannot be considered uncoupled from the interpolation (2.9). Also observe that at each grid level the integration has the fine grid solution $D_k^n R_{lk} U_l^{n-1}$ as initial function. Note that if we substitute the implicit Euler formula in (2.10), the scheme of [12] is obtained.

In (2.8) the approximations $U_k^{(i)}$ are defined on the whole of the grids ω_k and thus are also elements of S_k. Consequently, for any $k \geq 2$ interpolation is considered to take place on the whole of ω_k, which is costly. In actual application, the interpolations are therefore restricted to the nested integration domains. This point will be discussed later in the paper. For the time being, it is assumed that the numerical solutions are indeed generated as grid functions in S_k (grid expansion).

In (2.8) the number of grid levels l is fixed a priori, independent of time. In applications this fixed-level mode of operation may be inefficient. For example, if a solution steepens up in time, less levels are needed in the initial integration than at later times. Consequently, at early times l must be taken larger than necessary, which is not efficient. On the other hand, the solution may also become less steep, which again makes a fixed l inefficient. Obviously, the method should be capable of working with a variable l. For this variable-level mode of operation (2.8) requires a modification. Let l_{n-1}, l_n denote the number of levels from t_{n-1} to t_n and t_n to t_{n+1}, respectively. Then, for the step from t_{n-1} to t_n, (2.8) is modified to

$$U_1^{(i)} = R_{l_{n-1}1} U_{l_{n-1}}^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F_1(t_{n-1} + c_j \tau, U_1^{(j)}), \tag{2.11a}$$

$$1 \leq i \leq s+1 \; ; \quad k = 1,$$

$$U_k^{(i)} = D_k^n \, [R_{l_{n-1}k} U_{l_{n-1}}^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F_k(t_{n-1} + c_j \tau, U_k^{(j)})] + \tag{2.11b}$$

$$(I_k - D_k^n) \, [P_{k-1k} U_{k-1}^{(i)} + b_k^{(i)}], \quad 1 \leq i \leq s+1 \; ; \quad 2 \leq k \leq l_{n-1},$$

and, provided $l_n > l_{n-1}$, for $k = l_{n-1}+1, \cdots, l_n$ we have

$$U_k^{(i)} = P_{k-1k} U_{k-1}^{(i)} + b_k^{(i)}, \quad 1 \leq i \leq s+1. \tag{2.11c}$$

Consequently, if the number of levels should increase for use in the next step, then so-called full interpolations (2.11c) are carried out at the end of the current step, so that the required initial function, which is to be taken from the highest grid level that will be used, is always available. If $l_n \leq l_{n-1}$, then (2.11c) is omitted and nothing really changes. Full interpolation is necessary only when the solution steepens up in time. Because we will let the l_n depend exclusively on the spatial steepness, and because $\max_n \{l_n\}$ is finite, full interpolation is carried out only for a finite number of steps, uniformly in τ. Hence full interpolation cannot have a strongly diminishing effect on global accuracy. Like the matrices D_k^n, the actual choice for l_n is part of the adaptation strategy.

We conclude this section with a minor modification for certain RK methods. Above, D_k^n depends only on the step number n and the level index k, and not on the stages. There exist RK methods for which all coefficients a_{1j} are zero, trivially so for all explicit methods, but for example also for the implicit Lobatto IIIA-methods ($s = 2$ yields the familiar trapezoidal rule). If this is the case, then it is more natural to define for all grid levels the first-stage value as

$$U_k^{(1)} = R_{lk} U_l^{n-1} \tag{2.12}$$

to avoid interpolation. This means that at stage one D_k^n is to be replaced by the unit matrix I_k.

4.3. The general error scheme

To save space, (2.11) is rewritten as

$$\begin{aligned} U_k^{(i)} = D_k^n\, [R_{l_{n-1}k} U_{l_{n-1}}^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F_k(t_{n-1} + c_j\tau,\, U_k^{(j)})] + \\ (I_k - D_k^n)\, [P_{k-1k} U_{k-1}^{(i)} + b_k^{(i)}], \quad 1 \le i \le s+1\,; \quad 1 \le k \le l_n. \end{aligned} \tag{3.1}$$

Note that $D_1^n = I_1$ and $D_k^n = 0$ if $k > l_{n-1}$. Further, if $a_{1j} = 0$ $(1 \le j \le s)$, then D_k^n is to be replaced by I_k for $i = 1$, but only for $1 \le k \le l_{n-1}$. The rewriting of (2.11) into (3.1) introduces variables not existing in reality, viz. the grid functions $U_0^{(i)}$, $b_1^{(i)}$ and the operators P_{01} and $R_{l_{n-1}k}$ for $k > l_{n-1}$. Formally we can use (3.1) due to the definition of D_k^n.

The derivation of the error scheme parallels that in [12]. Consider the perturbed scheme

$$\begin{aligned} \tilde{U}_k^{(i)} = D_k^n\, [R_{l_{n-1}k} \tilde{U}_{l_{n-1}}^{n-1} + \tau \sum_{j=1}^{s} a_{ij} F_k(t_{n-1} + c_j\tau,\, \tilde{U}_k^{(j)})] + \\ (I_k - D_k^n)\, [P_{k-1k} \tilde{U}_{k-1}^{(i)} + b_k^{(i)}] + r_k^{(i)}, \quad 1 \le i \le s+1\,; \quad 1 \le k \le l_n, \end{aligned} \tag{3.2}$$

with the local perturbations $r_k^{(i)}$ still arbitrary. Introduce the errors

$$e_k^n = \tilde{U}_k^n - U_k^n, \quad e_k^{(i)} = \tilde{U}_k^{(i)} - U_k^{(i)}, \quad 1 \le i \le s+1\,; \quad 1 \le k \le l_n, \tag{3.3}$$

and subtract (3.1) from (3.2) to obtain

$$e_k^{(i)} = D_k^n \, [R_{l_{n-1}k} e_{l_{n-1}}^{n-1} + \tau \sum_{j=1}^{s} a_{ij} M_k^{(j)} e_k^{(j)}] + \tag{3.4}$$

$$(I_k - D_k^n) P_{k-1k} e_{k-1}^{(i)} + r_k^{(i)}, \qquad 1 \le i \le s+1 \, ; \qquad 1 \le k \le l_n .$$

$M_k^{(j)}$ is the integrated Jacobian matrix resulting from the use of the mean value theorem:

$$F_k(t_{n-1} + c_j \tau, \tilde{U}_k^{(j)}) - F_k(t_{n-1} + c_j \tau, U_k^{(j)}) = M_k^{(j)} (\tilde{U}_k^{(j)} - U_k^{(j)}), \tag{3.5a}$$

$$M_k^{(j)} = \int_0^1 F'(t_{n-1} + c_j \tau, \theta \tilde{U}_k^{(j)} + (1-\theta) U_k^{(j)}) d\theta. \tag{3.5b}$$

We next introduce the Kronecker product notation. Let E_{s+1} be the unit matrix of order $s+1$ and denote $e = [1, \cdots 1]^T \in \mathbb{R}^{s+1}$. Introduce the augmented vectors

$$\mathbf{e}_k^n = [e_k^{(1)\,T}, \ \cdots, e_k^{(s+1)\,T}]^T, \qquad \mathbf{r}_k^n = [r_k^{(1)\,T}, \ \cdots, r_k^{(s+1)\,T}]^T \tag{3.6}$$

in the augmented space $\mathbf{S}_k = \mathbb{R}^{(s+1)d_k}$ and the matrix operators

$$\begin{aligned} &\mathbf{R}_{l_{n-1}k} \colon \mathbf{S}_{l_{n-1}} \to \mathbf{S}_k, && \mathbf{R}_{l_{n-1}k} = E_{s+1} \otimes R_{l_{n-1}k} = \mathrm{diag}(R_{l_{n-1}k}), \\ &\mathbf{P}_{k-1k} \colon \mathbf{S}_{k-1} \to \mathbf{S}_k, && \mathbf{P}_{k-1k} = E_{s+1} \otimes P_{k-1k} = \mathrm{diag}(P_{k-1k}), \\ &\mathbf{I}_k \colon \mathbf{S}_k \to \mathbf{S}_k, && \mathbf{I}_k = E_{s+1} \otimes I_k = \mathrm{diag}(I_k). \end{aligned} \tag{3.7}$$

Define $\mathbf{D}_k^n \colon \mathbf{S}_k \to \mathbf{S}_k$, $\mathbf{D}_k^n = \mathrm{diag}(I_k, D_k^n, \ \cdots, D_k^n)$ if $a_{1j} = 0$, $1 \le j \le s$ (cf. (2.12)), and otherwise $\mathrm{diag}(D_k^n)$. Finally, we introduce the augmented Jacobian operators

$$\mathbf{M}_k^n = \begin{bmatrix} a_{11} M_k^{(1)} & a_{12} M_k^{(2)} & \cdots & a_{1s} M_k^{(s)} & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{s1} M_k^{(1)} & a_{s2} M_k^{(2)} & \cdots & a_{ss} M_k^{(s)} & 0 \\ a_{s+11} M_k^{(1)} & a_{s+12} M_k^{(2)} & \cdots & a_{s+1s} M_k^{(s)} & 0 \end{bmatrix}, \tag{3.8a}$$

$$\mathbf{Z}_k^n = \mathbf{I}_k - \tau \mathbf{D}_k^n \mathbf{M}_k^n, \tag{3.8b}$$

so that (3.4) can now be written in the compact form

$$\mathbf{Z}_k^n \mathbf{e}_k^n = \mathbf{D}_k^n \mathbf{R}_{l_{n-1}k}(e \otimes e_{l_{n-1}}^{n-1}) + (\mathbf{I}_k - \mathbf{D}_k^n)\mathbf{P}_{k-1k}\mathbf{e}_{k-1}^n + \mathbf{r}_k^n, \qquad (3.9)$$
$$1 \le k \le l_n.$$

In (3.9) we deal with an inner and outer recursion connected, respectively, with the grid refinement index k and time stepping index n. Introduce

$$\begin{aligned} \mathbf{X}_k^n &= (\mathbf{Z}_k^n)^{-1}(\mathbf{I}_k - \mathbf{D}_k^n)\mathbf{P}_{k-1k}, \\ \Gamma_k^n &= (\mathbf{Z}_k^n)^{-1}\mathbf{D}_k^n \mathbf{R}_{l_{n-1}k}, \\ \phi_k^n &= (\mathbf{Z}_k^n)^{-1}\mathbf{r}_k^n, \end{aligned} \qquad (3.10)$$

where $k = 1, \cdots, l_n$. Note that $\mathbf{Z}_k^n = \mathbf{I}_k$, $\mathbf{X}_k^n = \mathbf{P}_{k-1k}$, $\Gamma_k^n = \mathbf{0}$, $\phi_k^n = \mathbf{r}_k^n$ for the full interpolation levels $k = l_{n-1}+1, \cdots, l_n$. Using (3.10), (3.9) is rewritten as

$$\mathbf{e}_k^n = \mathbf{X}_k^n \mathbf{e}_{k-1}^n + \Gamma_k^n (e \otimes e_{l_{n-1}}^{n-1}) + \phi_k^n, \qquad k = 1, \cdots, l_n. \qquad (3.11)$$

An elementary calculation then leads to the final form

$$\mathbf{e}_k^n = \mathbf{G}_k^n (e \otimes e_{l_{n-1}}^{n-1}) + \psi_k^n, \qquad k = 1, \cdots, l_n, \qquad (3.12)$$

where $\mathbf{G}_k^n$ and ψ_k^n itself are also defined by recursions:

$$\mathbf{G}_1^n = \Gamma_1^n, \qquad \mathbf{G}_j^n = \mathbf{X}_j^n \mathbf{G}_{j-1}^n + \Gamma_j^n, \qquad j = 2, \cdots, k, \qquad (3.13)$$
$$\psi_1^n = \phi_1^n, \qquad \psi_j^n = \mathbf{X}_j^n \psi_{j-1}^n + \phi_j^n, \qquad j = 2, \cdots, k. \qquad (3.14)$$

Equation (3.12) describes the error propagation for increasing levels within one complete time step. When to be used as error recursion in time, we put $k = l_n$ as we use the highest level approximations $U_{l_{n-1}}^{n-1}$, $U_{l_n}^n$, $\cdots$ for step continuation. Hence,

$$\mathbf{e}_{l_n}^n = \mathbf{G}_{l_n}^n (e \otimes e_{l_{n-1}}^{n-1}) + \psi_{l_n}^n, \qquad n = 1, 2, \cdots, \qquad (3.15)$$

is the final error scheme for the highest level approximations. Similar as in the standard application of the RK method (single-level, multi-stage), our main interest concerns the $(s+1)$th component vector. Note that the formulation (3.15) supposes that $U_{l_n}^n$ is taken for output rather than $U_{l_{n-1}}^n$.

4.4. Remarks on stability

In [12] we have presented a comprehensive analysis of the stability of the multi-level implicit Euler method. The multi-level multi-stage RK formulas are not so feasible for a comprehensive stability analysis. A technical difficulty originates from the property that at any RK stage, nonphysical boundary values are defined by interpolating the solution of the corresponding stage from the next coarser grid. This implies that the internal RK stages play a role in the stability analysis, even for constant coefficient linear problems. On the other hand, we believe this role is little and that in application one encounters the same step-by-step stability as on a single grid, as long as interpolation takes place in low-error regions. In this paper no further attention is paid to stability analysis. Instead, we refer to the preprint [10] for some preliminary remarks on stability and proceed with the local error analysis which is to reveal how to define the adaptation strategy for choosing the spatial integration domains at the various refinement levels. Obviously, this is one of the main issues in the analysis, implementation and application of adaptive grid methods.

4.5. The local errors

4.5.1. Preliminaries

In the following, $||.||$ denotes the conventional maximum norm. We use the maximum norm as this norm is most natural for implementing adaptation strategies. Note that $||.||$ stands for the maximum norm in any space S_k or $\mathbf{S}_k$ under consideration, while the same symbol will be used for operators. We will examine the total local error ψ_k^n obtained by associating the local perturbations $\mathbf{r}_k^n$ with the true PDE solution. Note that the global errors $\mathbf{e}_k^n$ then become global discretization errors, viz.

$$\mathbf{e}_k^n = \mathbf{u}_k^n - \mathbf{U}_k^n, \qquad n = 0, 1, \ldots; \qquad 1 \le k \le l_n. \tag{5.1}$$

For clarity, henceforth we will consistently call ψ_k^n the total local error, whereas $\mathbf{r}_k^n$ will be consistently called a residual, so as to distinguish from ψ_k^n. Note that ψ_k^n can be interpreted as the k^{th}-level global error after one time step starting from the true PDE solution (put $e_{l_{n-1}}^{n-1} = 0$ in (3.12)).

We have tacitly used the natural assumption that any occurring augmented RK operator $\mathbf{Z}_k^n$ is invertible (under appropriate conditions on τ and ω_k). We thus may introduce the following bound:

$$||(\mathbf{Z}_k^n)^{-1}\mathbf{v}|| \le C||\mathbf{v}||, \quad \forall\, \mathbf{v} \in \mathbf{S}_k, \tag{5.2}$$

where $C \ge 1$ denotes a constant independent of τ and ω_k, while τ itself satisfies $\tau \le \tau_0$ with τ_0 possibly depending on $\omega_{l_{n-1}}$. The constant C and step size bound τ_0

are assumed to take on appropriate values (C close to 1 and τ_0 not unduly restrictive). As in [12], the aim of the error analysis is to derive a refinement condition that distributes space discretization and interpolation errors in such a way that the local spatial accuracy obtained on $\omega_{l_{n-1}}$ is comparable to the local spatial accuracy if this grid would be used without any adaptation. Assuming a stable time stepping process, this will then also be true for the global spatial accuracy.

4.5.2. The local error ψ_k^n

Replace, in the perturbed scheme (3.2), all $\tilde{U}_k^{(i)}$ values by the corresponding PDE solution values $u_k^{(i)}$. Then, in the space $\mathbf{S}_k$, the resulting residual $\mathbf{r}_k^n$ can be expressed as

$$\mathbf{r}_k^n = \mathbf{D}_k^n(\beta_k^n + \tau\sigma_k^n) + (\mathbf{I}_k - \mathbf{D}_k^n)\gamma_k^n, \tag{5.3}$$

where

$$\beta_k^n = [\beta_k^{(1)\,T}, \cdots, \beta_k^{(s)\,T}, \beta_k^{(s+1)\,T}]^T, \tag{5.4a}$$
$$\sigma_k^n = (A \otimes I_k)\,[\alpha_k^{(1)\,T}, \cdots, \alpha_k^{(s)\,T}, \alpha_k^{(s+1)\,T}]^T, \tag{5.4b}$$
$$\gamma_k^n = [\gamma_k^{(1)\,T}, \cdots, \gamma_k^{(s)\,T}, \gamma_k^{(s+1)\,T}]^T, \tag{5.4c}$$

The component $\beta_k^{(i)}$ is the PDE residual defined for the i^{th} RK stage:

$$\beta_k^{(i)} = u_k(t_{n-1} + c_i\tau) - u_k(t_{n-1}) - \tau\sum_{j=1}^{s} a_{ij}\frac{d}{dt}u_k(t_{n-1} + c_j\tau). \tag{5.5}$$

The component $\alpha_k^{(i)}$ is the PDE residual defined by the semi-discretization:

$$\alpha_k^{(i)} = \frac{d}{dt}u_k(t_{n-1} + c_i\tau) - F_k(t_{n-1} + c_i\tau, u_k(t_{n-1} + c_i\tau)). \tag{5.6}$$

Following common use, α_k^n and likewise σ_k^n and their components, will also be called local space discretization error. The matrix A represents the $(s+1)\times(s+1)$ Butcher matrix of RK coefficients a_{ij} whose $(s+1)^{\text{th}}$ column is zero. Hence, the i^{th} component $\sigma_k^{(i)}$ of σ_k^n is given by $\sigma_k^{(i)} = \sum_{j=1}^{s} a_{ij}\alpha_k^{(j)}$. Finally, the component $\gamma_k^{(i)}$ is the residual defined by the interpolation:

$$\gamma_k^{(i)} = u_k(t_{n-1} + c_i\tau) - P_{k-1k}u_{k-1}(t_{n-1} + c_i\tau) - b_k(t_{n-1} + c_i\tau) \tag{5.7}$$

and $\gamma_k^{(i)}$ and γ_k^n will also be called interpolation error. Observe that any component vector

$$r_k^{(i)} = D_k^n(\beta_k^{(i)} + \tau\sigma_k^{(i)}) + (I_k - D_k^n)\gamma_k^{(i)} \tag{5.8}$$

of $\mathbf{r}_k^n$ is now determined completely by the true PDE solution $u = u(\underline{x}, t)$. Thus, $r_k^{(i)}$ can be Taylor expanded assuming sufficient differentiability.

We are now ready to determine the local error ψ_k^n defined by recursion (3.14). Assuming

$$\prod_{i=k}^{k+1} \mathbf{X}_k^n = \mathbf{I}_k, \qquad k = 1, \cdots, l_n, \tag{5.9}$$

and using (3.10) and (5.3), we get

$$\psi_k^n = \sum_{j=1}^{k}(\prod_{i=k}^{j+1}\mathbf{X}_i^n)(\mathbf{Z}_j^n)^{-1}[\mathbf{D}_j^n(\beta_j^n + \tau\sigma_j^n) + (\mathbf{I}_j - \mathbf{D}_j^n)\gamma_j^n]. \tag{5.10}$$

A natural splitting into a temporal and a spatial local error is

$$\psi_k^n = \psi_{k,s}^n + \psi_{k,t}^n, \tag{5.11}$$

where

$$\psi_{k,s}^n = \sum_{j=1}^{k}(\prod_{i=k}^{j+1}\mathbf{X}_i^n)(\mathbf{Z}_j^n)^{-1}[\tau\mathbf{D}_j^n\sigma_j^n + (\mathbf{I}_j - \mathbf{D}_j^n)\gamma_j^n], \tag{5.12}$$

$$\psi_{k,t}^n = \sum_{j=1}^{k}(\prod_{i=k}^{j+1}\mathbf{X}_i^n)(\mathbf{Z}_j^n)^{-1}\mathbf{D}_j^n\beta_j^n. \tag{5.13}$$

The local space error $\psi_{k,s}^n$ contains only contributions from the spatial approximation, viz. local space discretization errors σ_j^n and spatial interpolation errors γ_j^n. The local time error $\psi_{k,t}^n$ contains only contributions β_j^n from the time integration. Hence, due to the splitting (5.11), for the spatial local error analysis we may restrict ourselves to $\psi_{k,s}^n$ and for the temporal local error analysis to $\psi_{k,t}^n$.

4.5.3. The local space error $\psi^n_{k,s}$

We rewrite $\psi^n_{k,s}$ as

$$\begin{aligned}\psi^n_{k,s} &= (\mathbf{Z}^n_k)^{-1}(\mathbf{I}_k - \mathbf{D}^n_k)\mathbf{P}_{k-1k} * \\ &\quad \sum_{j=1}^{k-1}(\prod_{i=k-1}^{j+1}\mathbf{X}^n_i)(\mathbf{Z}^n_j)^{-1}[\tau\mathbf{D}^n_j\sigma^n_j + (\mathbf{I}_j - \mathbf{D}^n_j)\gamma^n_j] + \\ &\quad (\mathbf{Z}^n_k)^{-1}[\tau\mathbf{D}^n_k\sigma^n_k + (\mathbf{I}_k - \mathbf{D}^n_k)\gamma^n_k] \\ &= (\mathbf{Z}^n_k)^{-1}[\tau\mathbf{D}^n_k\sigma^n_k + (\mathbf{I}_k - \mathbf{D}^n_k)\rho^n_k],\end{aligned} \tag{5.14}$$

where

$$\rho^n_1 = \mathbf{0} \quad \text{and} \quad \rho^n_k = \gamma^n_k + \mathbf{P}_{k-1k}\psi^n_{k-1,s}, \qquad k = 2, \cdots, l_n. \tag{5.15}$$

In (5.14), the local space discretization error $\mathbf{D}^n_k\sigma^n_k$, defined at the level-k integration domain, is separated from the local spatial error part $(\mathbf{I}_k - \mathbf{D}^n_k)\rho^n_k$ outside this domain. Note that ρ^n_k contains the level-k interpolation error γ^n_k and the prolongated local space error $\mathbf{P}_{k-1k}\psi^n_{k-1,s}$. At the full interpolation levels, (5.14) simplifies to

$$\psi^n_{k,s} = \gamma^n_k + \mathbf{P}_{k-1k}\psi^n_{k-1,s}, \qquad k = l_{n-1}+1, \cdots, l_n. \tag{5.16}$$

The separation of errors in (5.14) enables us to formulate the important refinement condition:

$$\|(\mathbf{Z}^n_{l_{n-1}})^{-1}(\mathbf{I}_{l_{n-1}} - \mathbf{D}^n_{l_{n-1}})\rho^n_{l_{n-1}}\| \le c\|(\mathbf{Z}^n_{l_{n-1}})^{-1}\tau\mathbf{D}^n_{l_{n-1}}\sigma^n_{l_{n-1}}\|, \tag{5.17}$$

where $c > 0$ denotes a threshold factor to be specified later. Substitution into (5.14) yields

$$\|\psi^n_{l_{n-1},s}\| \le (1+c)\|(\mathbf{Z}^n_{l_{n-1}})^{-1}\tau\mathbf{D}^n_{l_{n-1}}\sigma^n_{l_{n-1}}\|. \tag{5.18}$$

Hence, apart from the factor $(1+c)$, the local space error at the finest level is bounded by the local space discretization error on its integration domain. By imposing (5.17), we have virtually removed the error contribution from interpolation committed on all levels $k \le l_{n-1}$. Inequality (5.18) is in agreement with our goal of developing an adaptation strategy that generates integration domains in such a way that the spatial accuracy obtained on the finest level is comparable to that obtained without adaptation.

The refinement condition (5.17) implies constraints on the matrices $\mathbf{D}_k^n$ for $2 \le k \le l_{n-1}$. These constraints follow from the following derivation. Let, for brevity, $l = l_{n-1}$. With a simple calculation [12], we can rewrite ρ_l^n as

$$\rho_l^n = \lambda_l^n + \mathbf{P}_{l-1l} \sum_{k=2}^{l-1} \Big(\prod_{i=l-1}^{k+1} \mathbf{X}_i^n \Big) (\mathbf{Z}_k^n)^{-1} (\mathbf{I}_k - \mathbf{D}_k^n) \lambda_k^n, \tag{5.19}$$

where

$$\lambda_k^n = \gamma_k^n + \mathbf{P}_{k-1k} (\mathbf{Z}_{k-1}^n)^{-1} \tau \mathbf{D}_{k-1}^n \sigma_{k-1}^n, \qquad k = 2, \cdots, l, \tag{5.20}$$

contains the interpolation error at level k and the prolongated spatial discretization error of level $k-1$ to k (for $k = l - 1$ convention (5.9) applies). This λ-function will be used for determining the matrices $\mathbf{D}_k^n$. Let $C_I \ge 1$ be a constant such that

$$\|\mathbf{P}_{k-1k}\| \le C_I. \tag{5.21}$$

For linear interpolation $C_I = 1$, while for higher-order Lagrangian interpolation $C_I > 1$. Now,

$$\| \prod_{i=l-1}^{k+1} \mathbf{X}_i^n \| \le C_X \le (CC_I)^{l-k-1} \tag{5.22}$$

and using (5.19) we get

$$\|(\mathbf{Z}_l^n)^{-1} (\mathbf{I}_l - \mathbf{D}_l^n) \rho_l^n\| \le \tilde{C} \max_{2 \le k \le l} \{ \|(\mathbf{I}_k - \mathbf{D}_k^n) \lambda_k^n\| \} \tag{5.23}$$

with the grid independent constant

$$\tilde{C} = C(1 + C_I(l-2)CC_X) = C + (l-2)(CC_I)^{l-2} \tag{5.24}$$

Hence, if for each $k = 2, \cdots, l,$ the matrices $\mathbf{D}_k^n$ are selected such that

$$\|(\mathbf{I}_k - \mathbf{D}_k^n) \lambda_k^n\| \le \frac{c}{\tilde{C}} \|(\mathbf{Z}_l^n)^{-1} \tau \mathbf{D}_l^n \sigma_l^n\|, \qquad l = l_{n-1}, \tag{5.25}$$

then the refinement condition (5.17) is satisfied. In the remainder, (5.25) thus replaces (5.17).

This condition says that outside any integration domain the sum of the interpolation and prolongated spatial discretization error from the previous coarser level will be bounded by the spatial discretization error of the highest level, multiplied by $c/\tilde{C}$. This imposes a severe restriction on the size of the interpolation and discretization errors of the lower levels. On the other hand, this restriction is natural, because, when going to a higher level within the current time step, we never return to a grid point where the solution has been interpolated (nesting property). Note that in (5.25) the temporal step size τ features. In particular, if $\tau \rightarrow 0$, then the interpolation errors will prevail and $\mathbf{D}_k^n \rightarrow \mathbf{I}_k$. Recall that we interpolate at each time step so that interpolation errors can accumulate linearly with the number of time steps. Our refinement condition prevents this.

The refinement condition (5.25) is not applicable to the full interpolation levels as at these levels $\mathbf{D}_k^n = \mathbf{0}$. For simplicity, we now consider only one full interpolation level and note that this is sufficient for practical purposes. Using (5.16), if $l_n = l_{n-1}+1$ we thus find, instead of (5.18),

$$\|\psi_{l_n,s}^n\| \leq \|\gamma_{l_n}^n\| + C_I(1+c)\tau\|(\mathbf{Z}_{l_{n-1}}^n)^{-1}\mathbf{D}_{l_{n-1}}^n \sigma_{l_{n-1}}^n\|. \tag{5.26}$$

Recall that full interpolation occurs only in a finite number of steps, uniformly in τ. Hence, when adding all local errors for a convergence proof, assuming stability, this fact should be taken into account so as to avoid an overly pessimistic summation like

$$\sum_{j=1}^{n} \|\gamma_{l_j}^j\| \geq \frac{T}{\tau}\min_n \|\gamma_{l_n}^n\|. \tag{5.27}$$

With a more subtle summation, based on the finite number of full interpolations, the τ^{-1}-term is avoided.

In conclusion, by imposing the refinement condition (5.25), the local space error bounds (5.18), (5.26) are valid. In an implementation these bounds can be used to monitor the spatial accuracy, while (5.25) is then used for selecting the actual integration domains. Such an implementation is method dependent and therefore best to describe for a selected method. An illustration for a DIRK method is presented below. Finally, the error bound (5.18) suggests to choose the threshold factor c not too large. However, if we take c very small, then the effect will be that the greater part of the diagonal entries of $\mathbf{D}_k^n$ are put to unity to satisfy the refinement condition, which implies that the integration domains will become quite large.

4.5.4. The local time error $\psi^n_{k,t}$

Since the same τ is used at all levels, and β^n_k does not depend on the mesh width, we have $\beta^n_k = \mathbf{R}_{l_{n-1}k}\beta^n_{l_{n-1}}$ so that (5.13) yields

$$\psi^n_{k,t} = \sum_{j=1}^{k}\left(\prod_{i=k}^{j+1}\mathbf{X}^n_i\right)(\mathbf{Z}^n_j)^{-1}\mathbf{D}^n_j\mathbf{R}_{l_{n-1}j}\beta^n_{l_{n-1}}. \tag{5.28}$$

By comparison with the recursion (3.13) for the amplification operators G^n_k, one can see that

$$\psi^n_{k,t} = \mathbf{G}^n_k\beta^n_{l_{n-1}}, \qquad 1 \le k \le l_{n-1}. \tag{5.29}$$

This formula shows the dependence of the local time error on the temporal residual of the finest integration level. Alternatively, we may write, similar as for the local space error (5.14),

$$\psi^n_{k,t} = (\mathbf{Z}^n_k)^{-1}[\mathbf{D}^n_k\mathbf{R}_{l_{n-1}k}\beta^n_{l_{n-1}} + (\mathbf{I}_k - \mathbf{D}^n_k)\mathbf{P}_{k-1k}\psi^n_{k-1,t}], \tag{5.30}$$
$$k = 1, \cdots, l_{n-1}.$$

This representation shows more insight than (5.29). At each integration level we recover the local time error contribution committed on the integration domain, viz. $(\mathbf{Z}^n_k)^{-1}[\mathbf{D}^n_k\mathbf{R}_{l_{n-1}k}\beta^n_{l_{n-1}}]$, and the prolongation of the previous local time error of the next coarser level, viz. $(\mathbf{Z}^n_k)^{-1}[(\mathbf{I}_k - \mathbf{D}^n_k)\mathbf{P}_{k-1k}\psi^n_{k-1,t}]$.

Let $\tilde{p}$ denote the stage order of the RK method, [5, 8, 9]. Using (5.2) we have

$$\|\psi^n_{k,t}\| \le C\ \max\ \{\|\mathbf{D}^n_k\mathbf{R}_{l_{n-1}k}\beta^n_{l_{n-1}}\|, \|(\mathbf{I}_k - \mathbf{D}^n_k)\mathbf{P}_{k-1k}\psi^n_{k-1,t}\|\}. \tag{5.31}$$

Because both $\beta^n_{l_{n-1}}$ and $\psi^n_{1,t}$ are $O(\tau^{\tilde{p}+1})$, by definition of stage order, we thus trivially recover the usual stage-order result at all grid levels, that is,

$$\psi^n_{k,t} = O(\tau^{\tilde{p}+1}), \qquad 1 \le k \le l_{n-1}, \tag{5.32}$$

where, apart from the norm bounds C for $(\mathbf{Z}^n_k)^{-1}$ and C_I for $\mathbf{P}_{j-1j}$, the order constant involved depends exclusively on bounds for temporal derivatives of $u(\underline{x},t)$ (cf. (5.5)). To recover the conventional ODE order, p say, of the RK method, the $(s+1)$st output component of $\psi^n_{k,t}$ must be expanded. We then would also arrive at an order relation $\psi^n_{k,t} = O(\tau^{p+1})$, but here the constant involved may depend on negative powers of the mesh width, similar as in existing method-of-lines convergence theories (see [8, 9] and the preprint [10] on the order reduction phenomenon). Finally, no integration takes place at a full interpolation level, so that

$$\psi^n_{k,t} = \mathbf{P}_{k-1k}\psi^n_{k-1,t}, \qquad l_{n-1}+1 \le k \le l_n, \tag{5.33}$$

and we thus have the same temporal order as for $\psi^n_{k,t}$, $1 \le k \le l_{n-1}$.

4.6. Error analysis for a three-stage DIRK method

By way of illustration, in this section we elaborate the local error analysis for a three-stage DIRK method which later on will be used for presenting numerical examples.

4.6.1. The DIRK method

The DIRK method is found in [4] and defined by the Butcher array

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 2\theta & \theta & \theta & 0 \\ 1 & b_1 & b_2 & \theta \\ \hline & b_1 & b_2 & \theta \end{array} \qquad \begin{array}{l} \theta = (3+\sqrt{3})/6 \\ b_1 = 3/2-\theta-1/(4\theta) \\ b_2 = -1/2+1/(4\theta) \end{array} \tag{6.1}$$

It is strongly A-stable, has classical order $p = 3$, stage order $\tilde{p} = 2$, and uses only two effective stages (first row of coefficients is zero). Note that stage one and two define the trapezoidal rule and that stage three and four, the output stage, are identical.

4.6.2. Elaboration of the local time error

Assume, for simplicity, that the semi-discrete problem is of constant coefficient linear type,

$$\frac{d}{dt}U_k(t) = M_k U_k(t) + f_k(t). \tag{6.2}$$

Note that the linear case reveals the essentials of the local error analysis. Also for simplicity, we put $l_{n-1} = 2$. Conclusions for the higher-level case immediately follow. Thus, our task is to examine

$$\psi^n_{1,t} = (\mathbf{Z}^n_1)^{-1}\mathbf{R}_{21}\beta^n_2, \tag{6.3}$$

$$\psi_{2,t}^n = (\mathbf{Z}_2^n)^{-1}[\mathbf{D}_2^n\beta_2^n + (\mathbf{I}_2 - \mathbf{D}_2^n)\mathbf{P}_{12}\psi_{1,t}^n].$$

From (5.5), (6.1) we deduce $\beta_2^{(1)} = 0$, $\beta_2^{(4)} = \beta_2^{(3)}$ and

$$\beta_2^{(2)} = -\frac{2\theta^3}{3}\tau^3\frac{d^3}{dt^3}u_2(t_{n-1}) + O(\tau^4), \tag{6.4}$$
$$\beta_2^{(3)} = (\frac{1}{24} - \frac{\theta}{6} - \frac{\theta^2}{3} + \frac{2\theta^3}{3})\tau^4\frac{d^4}{dt^4}u_2(t_{n-1}) + O(\tau^5).$$

For any $\mathbf{v}_k \in \mathbf{S}_k$ having $\mathrm{v}_k^{(1)} = 0$, the components $\mathrm{w}_k^{(i)}$ of $\mathbf{w}_k = (\mathbf{Z}_k^n)^{-1}\mathbf{v}_k$ satisfy $\mathrm{w}_k^{(1)} = 0$,

$$\mathrm{w}_k^{(2)} = (I_k - \theta\tau D_k^n M_k)^{-1}\mathrm{v}_k^{(2)}, \tag{6.5}$$
$$\mathrm{w}_k^{(3)} = (I_k - \theta\tau D_k^n M_k)^{-2}b_2\tau D_k^n M_k \mathrm{v}_k^{(2)} + (I_k - \theta\tau D_k^n M_k)^{-1}\mathrm{v}_k^{(3)},$$

and $\mathrm{w}_k^{(4)} = w_k^{(3)}$. We note in passing that the bound (5.2) may be derived from

$$\|(I_k - \theta\tau D_k^n M_k)^{-1}\mathrm{v}_k\| \le (1-\theta\tau\mu)^{-1}\|\mathrm{v}_k\|, \qquad 1-\theta\tau\mu > 0, \tag{6.6}$$

with the logarithmic norm $\mu = \mu_\infty[D_k^n M_k]$ independent of (the mesh width of) M_k. This bound applies in all cases where implicit Euler integrates in a stable way [5, 12].

Now first put $k = 1$. In view of the foregoing we then find $\psi_{1,t}^{(1)} = 0$, $\psi_{1,t}^{(4)} = \psi_{1,t}^{(3)}$ and

$$\psi_{1,t}^{(2)} = -\frac{2\theta^3}{3}(I_1 - \theta\tau M_1)^{-1}\tau^3 R_{21}\frac{d^3}{dt^3}u_2(t_{n-1}) + O(\tau^4), \tag{6.7}$$
$$\psi_{1,t}^{(3)} = -b_2\frac{2\theta^3}{3}(I_1 - \theta\tau M_1)^{-2}M_1 R_{21}\tau^4\frac{d^3}{dt^3}u_2(t_{n-1}) + O(\tau^4).$$

Using the boundedness of the operators $(I_1 - \theta\tau M_1)^{-1}$, $(I_1 - \theta\tau M_1)^{-2}\tau M_1$, for $k = 1$ the stage-order result (5.32) with $\tilde{p} = 2$ is recovered. Also the classical order $p = 3$ follows from $\psi_{1,t}^{(4)}$ when interpreted as the local ODE error. However, then the order constant depends on $M_1 R_{21} d^3u_2/dt^3(t_{n-1}) = M_1 d^3u_1/dt^3(t_{n-1})$. Hence, $p = 3$ is meaningful only when $M_1 d^3u_1/dt^3(t_{n-1}) = O(1)$, uniformly in the mesh width, which is the case if the third derivative is zero at $\partial\Omega$. Otherwise the constant blows up for decreasing mesh width, making $p = 3$ not meaningful (order reduction, see [10] for a concrete example).

Next we put the level index $k = 2$. Since also $l_{n-1} = 2$, it suffices to examine the local error of the output stage, which is calculated from (6.3) as

$$\begin{aligned}
\psi_{2,t}^{(3)} &= (I_2 - \theta\tau D_2^n M_2)^{-2} b_2 \tau D_2^n M_2 [D_2^n \beta_2^{(2)} + (I_2 - D_2^n) P_{12} \psi_{1,t}^{(2)}] + \\
&\quad (I_2 - \theta\tau D_2^n M_2)^{-1} [D_2^n \beta_2^{(3)} + (I_2 - D_2^n) P_{12} \psi_{1,t}^{(3)}] \\
&= (I_2 - \theta\tau D_2^n M_2)^{-2} b_2 \tau D_2^n M_2 [D_2^n \beta_2^{(2)} + (I_2 - D_2^n) P_{12} \psi_{1,t}^{(2)}] + \qquad (6.8) \\
&\quad (I_2 - \theta\tau D_2^n M_2)^{-1} (I_2 - D_2^n) P_{12} \psi_{1,t}^{(3)} + O(\tau^4). \\
\psi_{2,t}^{(4)} &= \psi_{2,t}^{(3)}
\end{aligned}$$

Using boundedness of the operators, and the results for $k = 1$, stage order $\tilde{p} = 2$ directly follows. Inspection of the various terms also reveals the classical order $p = 3$. In connection with the occurrence of internal boundaries at grid interfaces, it is of interest to again examine the possibility of order reduction.

Distinguishing local error components outside and inside the integration domain, we can write,

$$(I_2 - D_2^n)\psi_{2,t}^{(4)} = (I_2 - D_2^n) P_{12} \psi_{1,t}^{(4)}, \tag{6.9a}$$

$$\begin{aligned}
D_2^n \psi_{2,t}^{(4)} &= (I_2 - \theta\tau D_2^n M_2)^{-2} b_2 \tau D_2^n M_2 \, * \\
&\quad [D_2^n \beta_2^{(2)} + (I_2 - D_2^n) P_{12} \psi_{1,t}^{(2)}] + \qquad (6.9b) \\
&\quad [(I_2 - \theta\tau D_2^n M_2)^{-1} - I_2](I_2 - D_2^n) P_{12} \psi_{1,t}^{(4)} + O(\tau^4).
\end{aligned}$$

Apart from the interpolation, the outside local error (6.9a) is completely determined by level-1 properties, so that a reduction at level 1 will also be felt at level-2 components outside the integration domain. The reduction will also be felt inside the level-2 integration domain, since (6.9b) depends on internal boundary values computed at level 1. An interesting question is, will the internal boundaries cause order reduction in case the physical one does not. To examine this question, we henceforth suppose that no reduction will take place at $\partial\Omega$ and thus assume the additional boundary condition $M_k d^3 u_k / dt^3 (t_{n-1}) = O(1)$, uniformly in the mesh width. Then $\psi_{1,t}^{(4)} = O(\tau^4)$, so that (6.8) yields

$$\begin{aligned}
\psi_{2,t}^{(4)} &= (I_2 - \theta\tau D_2^n M_2)^{-2} b_2 \tau D_2^n M_2 [D_2^n \beta_2^{(2)} + (I_2 - D_2^n) P_{12} \psi_{1,t}^{(2)}] + \\
&\quad O(\tau^4) \\
&= -b_2 \frac{2\theta^3}{3} (I_2 - \theta\tau D_2^n M_2)^{-2} \tau^4 D_2^n M_2 \, * \qquad (6.10) \\
&\quad [D_2^n \frac{d^3}{dt^3} u_2(t_{n-1}) + (I_2 - D_2^n) P_{12} \frac{d^3}{dt^3} u_1(t_{n-1})] + O(\tau^4).
\end{aligned}$$

Substitution of the interpolation error (5.7),

$$\gamma_2(t_{n-1}) = u_2(t_{n-1}) - P_{12} u_1(t_{n-1}) - b_2(t_{n-1}), \tag{6.11}$$

yields

$$\psi_{2,t}^{(4)} = b_2 \frac{2\theta^3}{3}(I_2 - \theta\tau D_2^n M_2)^{-2}\tau^4 D_2^n M_2 (I_2 - D_2^n)\frac{d^3}{dt^3}\gamma_2(t_{n-1}) \qquad (6.12)$$
$$+ O(\tau^4).$$

We note in passing that the additionally imposed boundary condition implies 'homogeneity in boundary conditions', causing the third derivative of $b_2(t)$ to vanish. From (6.12) we now deduce that if

$$\tilde{\gamma} \equiv D_2^n M_2 (I_2 - D_2^n)\frac{d^3}{dt^3}\gamma_2(t_{n-1}) = O(1), \qquad (6.13)$$

uniformly in the mesh width, then $\psi_{2,t}^{(4)} = O(\tau^4)$ uniformly in the mesh width.

Hence, assuming that at the physical boundary no order reduction takes place, an important conclusion is that the internal boundaries do not cause order reduction if the interpolation condition (6.13) holds. Fortunately, in applications this condition is easily satisfied. Sufficient is that

$$\|M_2\| \, \|\frac{d^3}{dt^3}\gamma_2(t_{n-1})\| = O(1), \qquad (6.14)$$

saying that the accuracy order of the interpolation should be greater than or equal to the spatial order of the differential operator (not to be confused with the order of consistency of the difference operator). For example, for second order in space problems it suffices to use simple linear interpolation.

4.6.3. Elaboration of the refinement condition

Given a specific integration method, the general refinement condition (5.25) needs to be simplified for practical use. Two main simplifications can be distinguished:

(i) The first has to do with the augmented form. Working with (5.25) requires computing in $\mathbf{S}_k$ which is expensive. Consequently, (5.25) is better replaced by an appropriate approximating condition in S_k, preferably connected with the output stage. It is always possible to carry this out, since the refinement condition is concerned with spatial errors. Apart from various multiplying bounded operators, these errors are similar over the stages.

Consider (5.20), (5.25). First we replace the Jacobian $M_k^{(i)}$ occurring in $\mathbf{Z}_k^n$ by an approximation M_k constant over the stages. M_k is taken to be the (approximate)

Jacobian, computed at the beginning of the time step. M_k is available as it is also used in the iterative Newton process for solving the implicit relations. Second, the augmented spatial error $\boldsymbol{\sigma}_k^n$ is approximated as

$$\boldsymbol{\sigma}_k^n = \begin{bmatrix} 0 \\ \theta\alpha_k^{(1)} + \theta\alpha_k^{(2)} \\ b_1\alpha_k^{(1)} + b_2\alpha_k^{(2)} + \theta\alpha_k^{(3)} \\ b_1\alpha_k^{(1)} + b_2\alpha_k^{(2)} + \theta\alpha_k^{(3)} \end{bmatrix} \approx \begin{bmatrix} 0 \\ 2\theta\alpha_k^{(3)} \\ \alpha_k^{(3)} \\ \alpha_k^{(3)} \end{bmatrix} \tag{6.15}$$

Note that we here truncate $O(\tau)$ terms and that $\alpha_k^{(3)} = \alpha_k^n = \alpha_k(t_n)$. Next, by using (6.5), the nontrivial components of the spatial error function $\mathbf{w}_k = (\mathbf{Z}_k^n)^{-1}\mathbf{D}_k^n\boldsymbol{\sigma}_k^n$ are approximated by

$$\begin{aligned} w_k^{(2)} &\approx 2\theta(I_k - \theta\tau D_k^n M_k)^{-1} D_k^n\alpha_k^n, \\ w_k^{(3)} &\approx (I_k - \theta\tau D_k^n M_k)^{-1}(2b_2\theta(I_k - \theta\tau D_k^n M_k)^{-1}\tau D_k^n M_k + I_k)D_k^n\alpha_k^n, \\ w_k^{(4)} &= w_k^{(3)} \end{aligned} \tag{6.16}$$

At each of the stages we recover a proportionality with the local space discretization error $D_k^n\alpha_k^n$. This justifies to select one particular stage. We choose the approximation

$$w_k^{(4)} \approx (1 - 2b_2)(I_k - \theta\tau D_k^n M_k)^{-1} D_k^n\alpha_k^n, \tag{6.17}$$

which avoids two forward-backward substitutions and is based on

$$[2b_2\theta(I_k - \theta\tau D_k^n M_k)^{-1}\tau D_k^n M_k + I_k]D_k^n\alpha_k^n \approx (1 - 2b_2)D_k^n\alpha_k^n. \tag{6.18}$$

In first approximation, (6.18) is exact if $D_k^n\alpha_k^n$ is taken to be an eigenvector belonging to the maximal eigenvalue. On the other hand, the operator in (6.18) is bounded, which justifies this step.

We now can replace the constituents of the regridding condition by their counterparts in S_k:

$$\|(I_k - D_k^n)\lambda_k^n\| \le \frac{c}{l-1}\|(1 - 2b_2)\tau(Z_l^n)^{-1}D_l^n\alpha_l^n\|, \tag{6.19}$$

$$k = 2, \cdots, l = l_{n-1},$$

$$\lambda_k^n = \gamma_k^n + (1 - 2b_2)\tau P_{k-1k}(Z_{k-1}^n)^{-1}D_{k-1}^n\alpha_{k-1}^n, \tag{6.20}$$

$$Z_k^n = I_k - \theta\tau D_k^n M_k. \tag{6.21}$$

Observe that $\|(I_k - D_k^n)\lambda_k^n\| = \|(\mathbf{I}_k - \mathbf{D}_k^n)\lambda_k^n\| + O(\tau)$. The choice $l-1$ for the constant $\tilde{C}$ is exact in case of linear interpolation, provided $C \le 1$ (see (5.24), (5.2)). We will use $\tilde{C} = l-1$ also in other situations and note that, apart from the constant $1-2b_2$, condition (6.19) is completely identical to the regridding condition found for the implicit Euler method in [12].

(ii) The second simplification has to do with the nesting property and restricted interpolation. Once at level $k-1$ the integration is completed, (6.19) is used to select the integration domain for level k. This selection process is carried out by the so-called flagging procedure which scans level-k points and flags those points for which (6.19) is violated to be placed within the new domain. Our mathematical framework prescribes that the scan be carried out on the whole of ω_k, as the interpolation error γ_k^n is defined on the whole of ω_k. This, of course, is time consuming. We therefore apply restricted interpolation, saying that the interpolation is restricted to level-k points lying within the $(k-1)^{\text{th}}$ integration domain. Subsequently, the scan is also restricted to the $(k-1)^{\text{th}}$ integration domain. This way the nesting of the integration domains is enforced. In [12] it is shown that restricted interpolation leads to (nearly) the same integration domains as found with full interpolation, hence full interpolation is truly redundant. Finally, the flagging procedure contains some safety measures (buffering) which enhances the reliability of the restricted interpolation. This procedure also implements numerical estimators for γ_k^n, $(Z_{k-1}^n)^{-1}D_{k-1}^n\alpha_{k-1}^n$ and $\|(Z_l^n)^{-1}D_l^n\alpha_l^n\|$. To save space, we again refer to [12].

4.7. Numerical examples

We will illustrate the outcome of the simplified refinement condition (6.19) of the DIRK method (6.1). Recall that, in theory, this condition guarantees local space errors at most equal to the maximum of the local space error on the finest grid when used without adaptation, up to a certain grid independent constant (arising, e.g., from transferring the refinement condition to S_k and estimating $\tilde{C}$ by $l-1$). Hence, assuming stability, our theory dictates that the usual convergence behavior of the discretization method applied without adaptation will be maintained.

Two examples are presented, both 2D. The first serves to illustrate the above claim on convergence. This problem is solved using the 'fixed-level mode of operation'. The second serves to illustrate the performance of the method when applied in the 'variable-level mode of operation'. This mode is advocated if the solution shape strongly changes in time, e.g., when steep layers emerge at later times and at earlier times large gradients are absent. In such situations it is important that new levels are created in time in order to preserve accuracy. On the other hand, new levels should not be created too early for efficiency.

4.7.1. Example problem I.

The equation is linear and parabolic and given by (*Adjerid* and *Flaherty* [1])

$$u_t = u_{xx} + u_{yy} - u_x - u_y + f(x,y,t), \qquad 0 < x, y < 1, \qquad t > 0. \tag{7.1}$$

The initial and Dirichlet boundary conditions and forcing function f are adjusted to

$$u(x,y,t) = 1 - \tanh(25(x-t) + 5(y-1)). \tag{7.2}$$

This solution is a skew wave propagating through the domain from left to right. The wave starts near the left boundary and approaches the right boundary at approximately $t = 0.8$. We integrate over the time interval [0,0.6]. This problem is suitable to subdue the LUGR method to a convergence test.

The spatial discretization is based on second-order symmetric differences. Simple linear interpolation is used and the constant c, introduced in the refinement condition, is put equal to one. Four computations were performed using, respectively, 1,2,3 and 4 levels. The mesh width in both x- and y-direction of the base grid is 0.05. During a computation the step size τ is fixed. However, when adding a level, we simultaneously halve τ. Because the stage order of the DIRK method is 2, like the order of the spatial discretization, per computation a gain factor of approximately 4 then should be found for the total global errors. To compare the accuracy with the accuracy obtained on a single, uniform grid, we have also solved the problem in the standard way using the same values for τ and the mesh width of the finest level. The values for τ and the mesh width in space are always such that the space error dominates. For illustration purposes this is necessary, since otherwise no valid conclusion can be drawn on the performance of the spatial refinement condition.

The results of the computations are contained in Table 7.1. We see that the LUGR solutions converge according to the theory and, also, that these solutions are as accurate as the standard, uniform grid solutions. In view of the simplifications of Section 6.3, this correspondence in accuracy is striking. We should note, though, that in the actual flagging procedure some safety measures have been built, like buffering. Buffering of course helps in keeping the LUGR accuracy close to the standard accuracy. Figure 7.1 shows the grids of the 2-,3- and 4-level computations at two different times. Note that the grids align with the wave front and become larger for smaller τ, in accordance with (6.19).

4.7.2. Example problem II.

The equation is again linear and parabolic,

$$u_t = u_{xx} + u_{yy} + f(x,y,t), \qquad 0 < x, y < 1, \qquad t > 0. \tag{7.3}$$

τ	# of levels	single grid	t 0.3	0.6
0.1	1	20x20	0.17319	0.17401
0.05	2		0.02728	0.02815
		40x40	0.02789	0.02810
0.025	3		0.00624	0.00716
		80x80	0.00680	0.00684
0.0125	4		0.00177	0.00174
		160x160	0.00168	0.00169

TABLE 7.1. Example problem I. Maxima of global errors computed at the finest available level. Comparison with the accuracy obtained on a single, uniform grid.

The initial and Dirichlet boundary conditions and forcing function f are adjusted to

$$u(x,y,t) = 1 - \tanh(100[(x-0.5)^2 + (y-0.5)^2 - t + 0.025]). \qquad (7.4)$$

This solution rapidly varies its shape and serves to illustrate the 'variable-level mode of operation'. At $t = 0$ the solution is almost zero over the entire domain. As time elapses it steepens up at [0.5,0.5], developing a circular wave front. This front starts to propagate towards the boundaries when $u(0.5,0.5,t) \approx 2$ and during the propagation the front becomes steeper. When the front has passed a point (x,y), the solution $u(x,y,t)$ approximates the value 2. We solve the problem over the time interval [0,0.1], which is sufficiently large to see all phenomena happen.

The refinement condition (6.19) tells us where to integrate on a finer level. When using the 'fixed-level mode of operation' this suffices. When using the 'variable-level mode of operation', we also need a criterion to decide when to change the number of levels. A natural thing to do is to associate this criterion to the spatial local error value. In the present experiment we employ the numerical spatial local error expression as used in (6.19). Within each base time step we monitor the number of grid levels with the criterion

$$(1+c)(1-2b_2)\tau\|(Z_k^n)^{-1} D_k^n \alpha_k^n\| < \tau\mathrm{TOL}, \qquad (7.5)$$

where TOL represents a tolerance value. Starting with $k = 1$, this inequality is checked after each level integration. If it is violated, then k is increased by 1. Otherwise it is decided that enough levels have been introduced and l_{n-1} is assigned the current value for k. Hence, the idea is to select l_{n-1} in such a way that the local

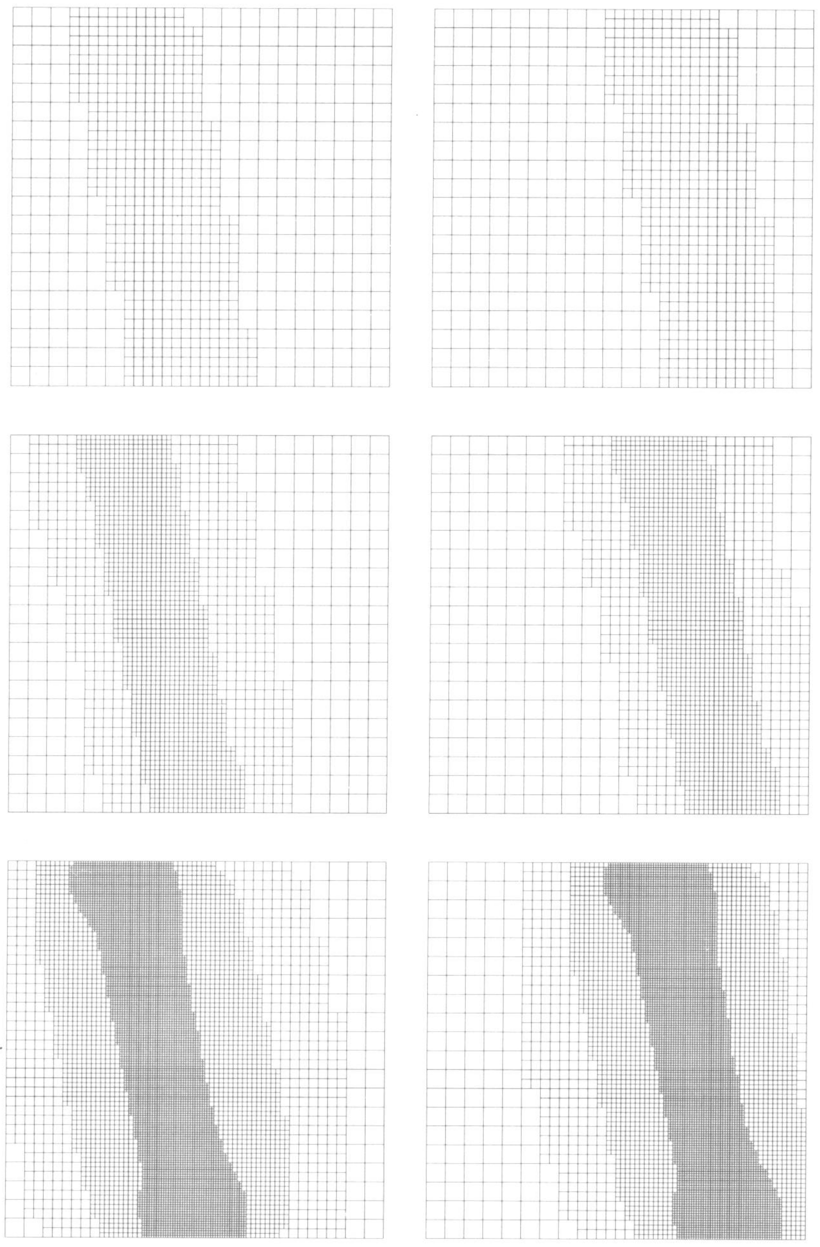


FIGURE 7.1. Example Problem I. Grids of the 2-,3- and 4-level computations at $t = 0.3$ and $t = 0.6$.

error expression in (7.5) is kept close to τTOL.

We will encounter a few full interpolations. The full interpolation error is neglected in (7.5). We justify this heuristic decision with the observation that full interpolation can take place only in a few number of steps (see also Section 5). However, to remain on the safe side, we now use fourth-order Lagrangian interpolation instead of second-order linear. It is obvious that full interpolation should not diminish the quality of the approximations, since otherwise the estimation of the discretization and interpolation errors used by the refinement condition is hindered. The full interpolation should also not interfere with the estimation of the number of levels needed in the step to follow. Therefore, the additional errors stemming from full interpolation have to be restricted in some manner. In the present experiment fourth-order interpolation has turned out to work satisfactorily.

t	# of levels	global error
0.01	1	0.01074
0.017		0.03171
0.018	2	0.01222
0.02		0.01117
0.03		0.01612
0.039		0.02523
0.04	3	0.01392
0.05		0.01493
0.06		0.01668
0.07		0.02168
0.072		0.02136
0.073	4	0.01289
0.08		0.01191
0.09		0.00722
0.1		0.00713

TABLE 7.2. Example problem II. Maxima of global errors computed at the finest grid at various time points, including those where a new grid is introduced.

The actual experiment with problem (7.3) - (7.4) concerns one run over the time interval [0,0.1]. The constant c of the refinement condition is again put equal to 1. The step size $\tau = 0.001$ and is kept constant. The value of 0.001 is sufficiently small to guarantee that spatial effects dominate. The mesh width in both x- and y-direction of the base grid is 0.05 and the tolerance parameter TOL = 50. Results are collected in Tables 7.2-7.3 and Figure 7.2. For a subset of time points, including those where a new grid level is added, Table 7.2 shows the course of the number of grid levels and the maximum of the global error measured at the finest available grid. Note that while the circular wave front develops, the algorithm keeps the error at a fairly constant level, which is in line with the idea behind the error monitor (7.5).

t	level	approx.	exact
0.017	1	0.04616	0.05307
0.018	1	0.05165	0.05793
	2	0.01246	0.01468
0.039	1	0.12511	0.28075
	2	0.04678	0.05579
0.04	1	0.13972	0.33177
	2	0.05084	0.06329
	3	0.01495	0.01467
0.072	1	0.31630	1.48171
	2	0.18625	0.30880
	3	0.04685	0.05130
0.073	1	0.34144	1.39739
	2	0.18078	0.29537
	3	0.05088	0.05367
	4	0.01683	0.01382

TABLE 7.3. Example problem II. Exact values and numerical estimates of the spatial local error expression (7.5). Note that the step size $\tau = 0.001$ is contained in these values.

The pictures contained in Figure 7.2 illustrate that the grids accurately reflect the circular wave front form (symmetry), showing that the refinement condition, which tells us where to refine, works as anticipated. On the other hand, the number of levels needed is not always computed optimally. This happens, e.g., at $t = 0.04$ and $t = 0.073$, time points where a new grid level is used for the first time. The grid pictures show that at these time points the new fine grid almost completely overlaps the existing one, indicating that the new fine grid is introduced too late (the solution steepens up). Fortunately, Table 7.2 shows that this small deficiency does not diminish the accuracy for evolving time. Also note that at later points of time this phenomenon disappears, see $t = 0.05$ and $t = 0.1$. This is of course what should happen due to the ever increasing solution gradients.

The precise origin of this small deficiency is not clear. The error introduced by the full interpolation can play a role here (this error is not monitored by (7.5)). More likely is, however, that it emanates from the lack of asymptotics at the coarser grids. This lack of asymptotics is inherent to any monitoring process that starts on coarse grids and therefore very difficult to overcome. To provide insight in the asymptotics for the estimator of (7.5), we have added Table 7.3. This table shows the exact, analytical values for (7.5) with their estimated numerical values at time points just

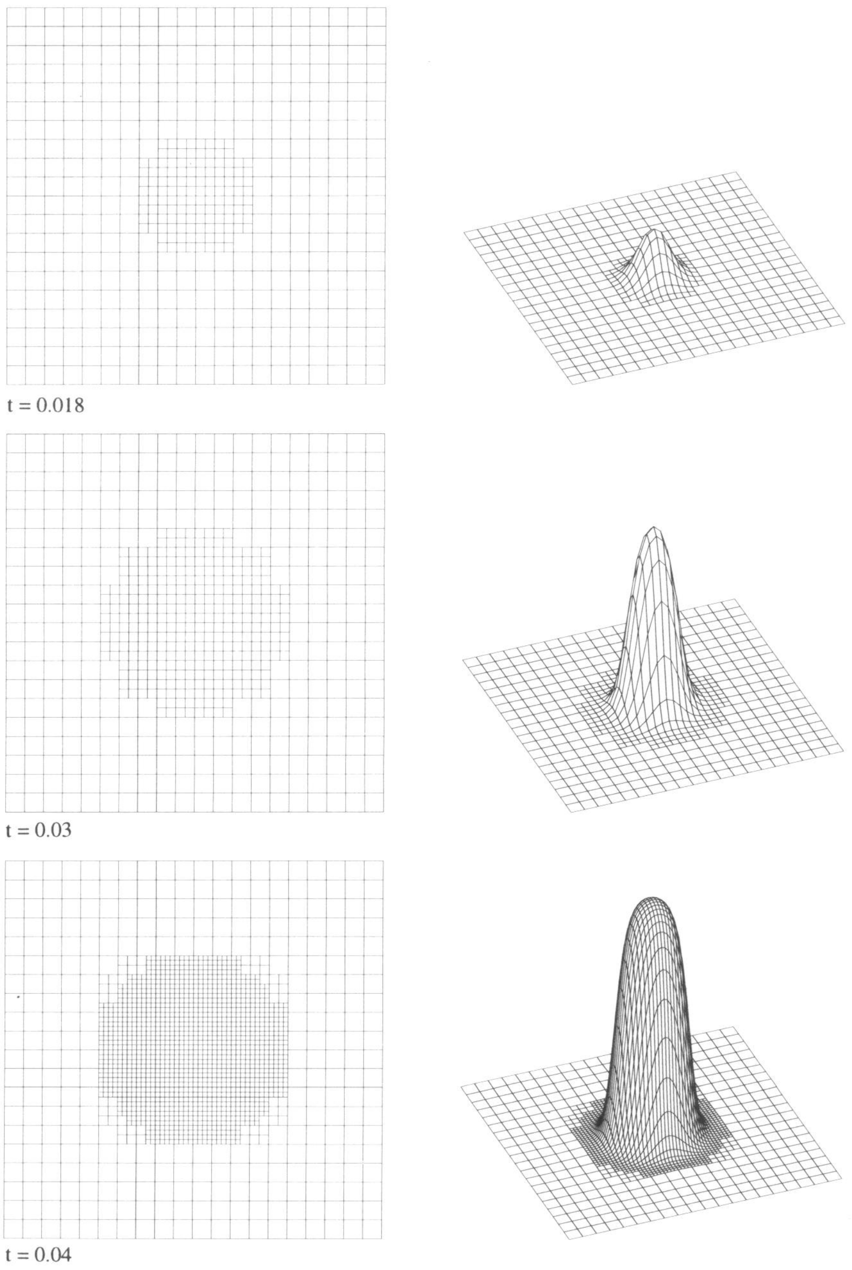


FIGURE 7.2. Example Problem II. The course of the local, uniform grids and the computed solution of (7.3).

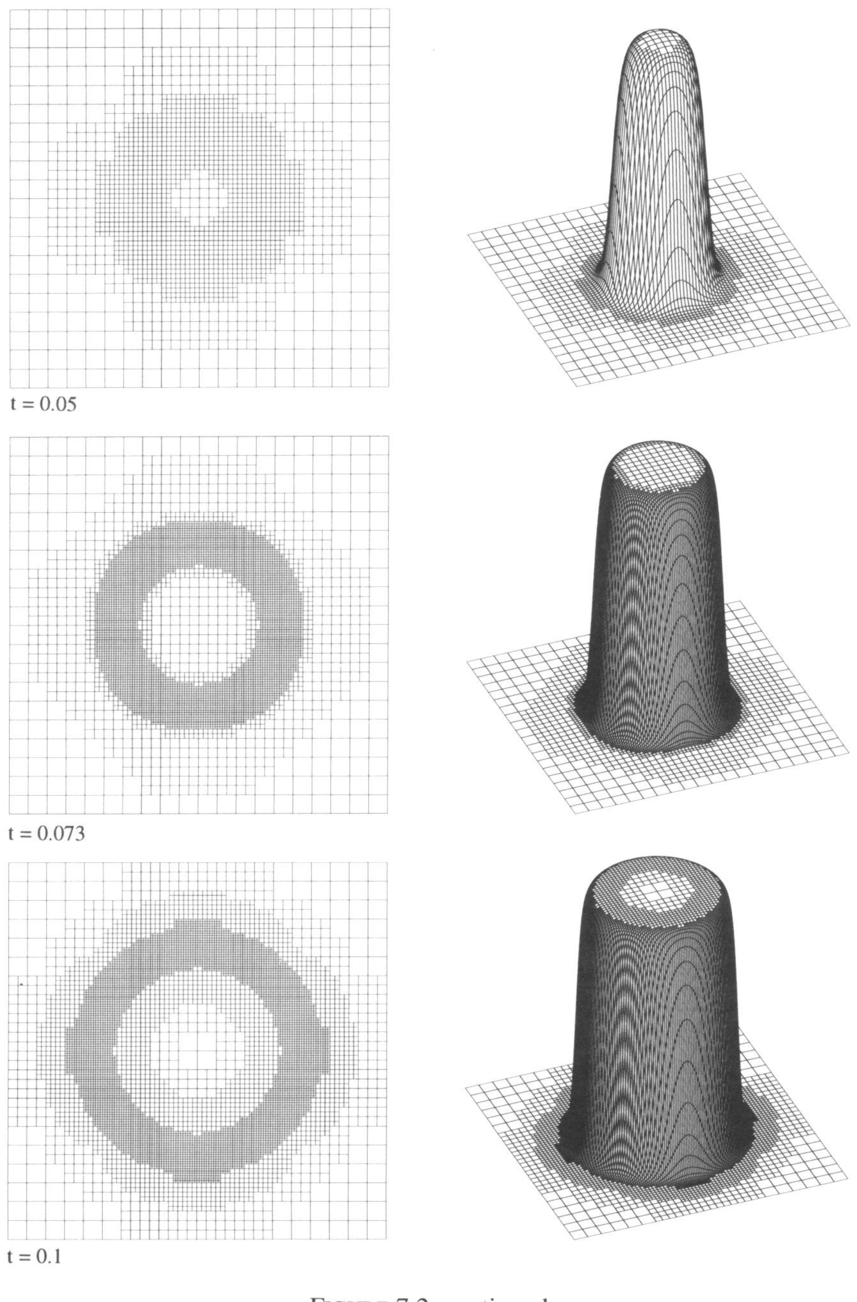


FIGURE 7.2. continued.

before and after the introduction of a new grid level. First, we see that at corresponding levels before and after the listed time points the numerical estimations are in fairly good agreement with one another, even on the coarse base grid. This supports the conclusion that the full interpolation is sufficiently accurate for not interfering with the selection of number of levels. Second, there is excellent agreement between the exact and numerical values on the fine grids. However, particularly at later times, the coarse grid values are not in good agreement with one another. This means that we are outside the asymptotic regime and this is likely to cause some disturbances in the selection of the right number of levels. We wish to emphasize once more that in spite of this lack of asymptotics, the overall behavior of the algorithm is very satisfactorily.

Let us conclude with a remark on the choice of TOL, in connection with the discrepancy between the value TOL = 50 and the global accuracy shown in Table 7.2. A discrepancy like this is unavoidable, due to damping of global errors. Note that we have a parabolic problem and that the DIRK method mimics the damping property of the parabolic operator (strong A-stability). Part of the discrepancy may also originate from cancellation between temporal and spatial terms. This damping of global errors, and this eventual cancellation, has not been taken into account in our error analysis which focuses on local errors, in particular on local error bounds. For precise estimation purposes our analysis is simply too general. On the other hand, the present example once more shows that local error bounds like (7.5) can be much too conservative (the simplified form is not essential for the present discussion). Consequently, for application, local error expressions like (7.5) are better be interpreted as error monitors. In connection with grid selection purposes, our practical experience is that with this interpretation the (simplified) spatial local error expression is reliable and works very satisfactorily.

4.8. Efficiency of time stepping

An important subject for future research is that of efficiency of the time-stepping scheme itself when combined with the LUGR technique. Two important issues not addressed in this work concern the use of variable time steps and the solution of the arising systems of linear and nonlinear algebraic equations, in case of an implicit scheme. Straightforward use of variable time step algorithms, as successfully applied in single-grid, method-of-lines computations, renders problems due to the fact that approximations obtained with an LUGR method are always difficult to numerically differentiate in time. The reason for this is that part of the components is obtained from a numerical integration, part from interpolation or injection. The resulting 'nonsmoothness' is then felt when computing higher temporal derivatives. More precisely, the higher temporal derivatives are estimated in a rough way, resulting in disturbances in the step size selection (see also [11]). To our experience, smoothing or filtering procedures provide only a partial remedy here.

Concerning the second issue, by nature of the LUGR approach approximations are computed in varying dimensions, even within one base time step. For DIRK or alternative implicit methods this obviously implies that the numerical algebra effort, to be made in solving systems of algebraic equations, becomes highly important. In the numerical experiments reported here, we have paid no attention to the efficiency of the numerical algebra computations and simply used an available sparse matrix technique (same as in [12]). This technique, however, is known to yield a considerable overhead when used in the solution of time-dependent problems. It is most likely that sophisticated iterative solution procedures will be much more effective.

REFERENCES

1. S. ADJERID and J.E. FLAHERTY (1988). A local Refinement Finite Element Method for Two Dimensional Parabolic Systems, *SIAM J. Sci. Statist. Comput.*, 9, 792-811.
2. D.C. ARNEY and J.E. FLAHERTY (1989). An Adaptive Local Mesh Refinement Method for Time-Dependent Partial Differential Equations, *Appl. Numer. Math.*, 5, 257-274.
3. M.J. BERGER and J. OLIGER (1984). Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, *J. Comput. Phys.*, 53, 484-512.
4. M. CROUZEIX and P.A. RAVIART (1980). *Approximation des Problèmes d'Evolution. Première Partie: Etude des Méthodes Linéaires a Pas Multiples et des Méthodes de Runge-Kutta,* Unpublished Lecture Notes, Université de Rennes, France.
5. K. DEKKER and J.G. VERWER (1984). *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, North-Holland, Amsterdam-New York-Oxford.
6. W.D. GROPP (1987). Local Uniform Mesh Refinement on Vector and Parallel Processors, in *Large Scale Scientific Computing*, 349-367, ed. P. DEUFLHARD, B. ENGQUIST, Birkhäuser Series Progress in Scientific Computing.
7. W.D. GROPP (1987). Local Uniform Mesh Refinement with Moving Grids, *SIAM J. Sci. Statist. Comput.*, 8, 292-304.
8. J.M SANZ-SERNA and J.G. VERWER (1989). Stability and Convergence at the PDE/Stiff ODE Interface, *Appl. Numer. Math.*, 5, 117-132.
9. J.M SANZ-SERNA, J.G. VERWER, and W.H. HUNDSDORFER (1987). Convergence and Order Reduction of Runge-Kutta Schemes Applied to Evolutionary Problems in Partial Differential Equations, *Numer. Math.*, 50, 405-418.
10. R.A. TROMPERT and J.G. VERWER (1990). *Runge-Kutta Methods and Local Uniform Grid Refinement,* NM-R9022, Centre for Mathematics and Computer Science, Amsterdam.
11. R.A. TROMPERT and J.G. VERWER (1991). A Static-Regridding Method for Two Dimensional Parabolic Partial Differential Equations, *Appl. Numer. Math.*, 8, 65-90.
12. R.A. TROMPERT and J.G. VERWER (1993). Analysis of the Implicit Euler Local Uniform Grid Refinement Method, *SIAM J. Sci. Comput.*, 18, 259-278.

Chapter 5

Local Uniform Grid Refinement and Systems of Coupled Partial Differential Equations

Ron Trompert
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

In this paper we consider an adaptive grid method based on local uniform grid refinement applied to systems of coupled time-dependent partial differential equations (PDEs). Local uniform grid refinement means that the PDEs are solved on a series of nested, uniform, increasingly finer subgrids which cover only a part of the domain. These subgrids are created up to a level of refinement where sufficient spatial accuracy is obtained and their location and shape is adjusted after each time step in order to follow the moving steep fronts. When a system of coupled PDEs is solved, the behavior of the local and global error associated with each separate PDE can be very different from one PDE to another. A refinement strategy based on a global error analysis has been developed which takes these differences into account. This refinement strategy aims at the domination of the global space error by the space discretization error at the finest subgrid.

published in: Appl. Numer. Math. **12** (1993), 331-355

5.1. Introduction

The local uniform grid refinement method is an adaptive grid method used to solve time-dependent partial differential equations (PDEs) with locally steep solutions. For such problems, a uniform space grid can be computationally very inefficient, since, to obtain an accurate approximation, such a grid would easily have to contain an excessive number of nodes, particulary so in two and three space dimensions.

The main feature of local uniform grid refinement is that the PDEs are solved on a series of nested, uniform, Cartesian, increasingly finer subgrids covering only that part of the domain where the spatial error is high. The PDEs are solved on each separate subgrid in a consecutive manner, from coarse to fine. The location and size of the subgrids are automatically adjusted at discrete times in order to follow the movement of the steep fronts. The generation of subgrids is continued until sufficient spatial accuracy is reached.

So far, local uniform grid refinement methods were proposed in a number of different varieties and applied to different kinds of PDEs. Here, we will not attempt to give a complete overview of the field. We will only sketch some varieties of the local uniform grid refinement method briefly and provide some references for interested readers. The methods contained in [1-4] are applied to hyperbolic PDEs and use explicit time stepping techniques. The method proposed by *Berger* and *Oliger* in [3] employs subgrids which are rectangles which may be skewed with respect to the co-ordinate axes in order to align with the steep region of the solution. Subgrids having the same cell sizes can partially overlap each other in this method. In [1], *Arney* and *Flaherty* developed a method very similar to the one in [3] except that the subgrids here are created by cellular refinement, meaning that the fine grid cells are properly nested within coarser grid cells. Hence, these subgrids have a piecewise polygonal shape.

Local uniform grid refinement is combined with grid movement in [2,4]. In [4], a method proposed by *Gropp* uses subgrids which are rectangles having sides parallel to the co-ordinate axes and which are able to move as a whole with the moving steep fronts. In this method the subgrids are also allowed to overlap each other. In [2], *Arney* and *Flaherty* added grid movement to their method described in [1]. The grid nodes of the coarsest grid are able to move and the fine grid movement is induced by the movement of the coarsest grid. Local uniform grid refinement methods are also used to solve parabolic and elliptic PDEs in [5,6] and involve the implicit solution of systems of equations. The subgrids in [6] are piecewise polygonal and the ones in [5] are rectangles. In both [5,6] domain decomposition is applied to improve the performance on parallel computers.

Our previous work on this type of adaptive grid method is contained in [7-11]. The subgrids in our method have a piecewise polygonal shape and do not overlap. Our method is a static-regridding method which means that no grid movement is applied during a time step. The refinement strategy controlling the generation of subgrids in [7] is based on heuristic criteria while in [8-11] it is underlied by a comprehensive error analysis which has resulted in a so-called refinement condition. This condition has been designed so that when this condition is satisfied during the grid refinement process and the number of subgrids is fixed in time, then the spatial accuracy of the solution obtained with the adaptive grid method should be comparable to the spatial accuracy obtained using one uniform grid covering the entire spatial domain when the cell sizes of this uniform grid are identical to those of the finest subgrid in use in the adaptive grid method. The refinement strategy is designed to fulfill the refinement condition. Due to the refinement condition a convergence result as if a single, uniform grid was used could be proved in certain model situations. The error analysis was carried out for the local uniform grid refinement method applied to time-dependent PDEs which after spatial discretization yield a system of ordinary differential equations (ODEs). However, when a system of coupled PDEs is solved, this need not be the case. It is known that the global and local error components associated with each separate PDE belonging to such a system can behave differently from one PDE to another. This means that, for example, the global error corresponding with one PDE can propagate in a different way to future time levels than the one associated with another PDE. With respect to the local error, this difference in behavior means that the local errors connected with different PDEs do not always behave in the same way when the time step size tends to zero. For this reason the refinement strategy has to be adapted to this more general class of PDEs. Moreover, in most of our previous work, the refinement strategy is aimed at controlling the spatial accuracy or global space error by, in some sense, controlling the local space error. This strategy performs satisfactorily but can be very restrictive, especially when the number of subgrids is large or the time step size very small. In this paper, the error analysis is redone for systems of coupled PDEs. From this, a more general and much less restrictive refinement strategy is obtained aiming at controlling the spatial accuracy by estimating the global space error itself.

In Section 2 a brief outline will be given of our version of the local uniform grid refinement method. Section 3 deals with the mathematical formulation of the method needed for the error analysis. The results of the error analysis are given in Section 4. This section also considers the influence of a system of coupled PDEs on the behavior of the global and local error. The refinement strategy is discussed in Section 5. Three example problems were used to illustrate the performance of the method. The results of these tests are given in Section 6. Although the example problems involve two space dimensions, the error analysis, refinement condition and refinement strategy applies to any number of space dimensions. The final Section 7 contains the summary and concluding remarks.

5.2. Outline of the adaptive grid method

Although its elaboration readily becomes complicated, the idea behind local uniform grid refinement is simple. Starting from a coarse grid, finer-and-finer uniform subgrids are created locally in a nested manner in regions where the solution is steep. Here, a set of interconnected grid cells, all having the same sizes, is called a *subgrid*. A set of subgrids having the same cell sizes is called a *grid level* or just *grid*. Hence, a grid level consists of a single subgrid or several disjunct non-overlapping subgrids. A new (initial-) boundary value problem is solved at each grid level in a consecutive order, from coarse to fine using the same time step size for all grid levels. This means that the refinement in time is global, i.e. the step size is adapted in time but is the same for all grid levels in use. Note that the PDEs are solved on a grid level as a whole, in spite of the fact that the grid level can consist of several disjunct subgrids. The required initial values for the finer subgrids are defined by interpolation from the coarser subgrid or taken from a finer subgrid from the previous time step when available. Internal boundaries, i.e. subgrid boundaries lying in the interior of the domain, are treated as Dirichlet boundaries and values are also interpolated from the next coarser grid level. Where the boundary of a fine subgrid coincides with the boundary of the domain, the prescribed boundary conditions are used. Except for the necessary initial and boundary conditions, all subgrids are independent of each other. Therefore, the subgrids are not patched into the coarser grids but are actually overlaying them. The generation of subgrids is continued until the spatial phenomena are described accurate enough by the finest grid. The fine grid cells are created by bisecting the sides of the cells of the next coarser grid, so the refinement is cellular. The subgrids created this way have a piecewise polygonal shape. Further, the unknowns are defined at cell vertices which implies that in the region where the coarse grid is overlapped by the fine grid, the coarse grid nodes coincide with the fine grid nodes.

During each time step the following operations are performed:

(1) *Solve the PDEs on the coarsest grid level.*
(2) *If the desired accuracy in space or the maximum number of grid levels is reached then go to 8.*
(3) *Determine new finer uniform grid level at forward time.*
(4) *Interpolate internal boundary values at forward time.*
(5) *Provide new initial values at backward time.*
(6) *Solve the PDEs on the new grid level, using the same step length.*
(7) *go to 2.*
(8) *Assign fine grid values to the corresponding coarser grid points.*

Thus, for each time step the computation starts at the coarse base grid using the most accurate solution available, since coarse grid solution values are always replaced by fine grid values at coarse grid nodes coinciding with fine grid nodes and all subgrids are kept in storage for step continuation.

5.3. Mathematical formulation

A mathematical formulation will be given needed for the error analysis of the local uniform grid refinement method. The following system of PDEs is considered, together with the initial and boundary conditions, defined on a domain Ω in $\mathbb{R}$, $\mathbb{R}^2$ or $\mathbb{R}^3$ with boundary $\partial\Omega$ and sides parallel to the co-ordinate axes,

$$\mathcal{A}_\Omega\,(\underline{x},t,u)\,u_t = \mathcal{F}_\Omega(\underline{x},t,u), \quad t > t_0, \quad \underline{x} \in \Omega, \tag{3.1a}$$

$$\mathcal{A}_{\partial\Omega}\,(\underline{x},t,u)\,u_t = \mathcal{F}_{\partial\Omega}\,(\underline{x},t,u), \quad t > t_0, \quad \underline{x} \in \partial\Omega, \tag{3.1b}$$

$$u(\underline{x},t_0) = u_0(\underline{x}), \quad \underline{x} \in \Omega\cup\partial\Omega. \tag{3.1c}$$

This system of PDEs is assumed to possess a unique solution $u(\underline{x},t)$, which is as often differentiable as the numerical analysis requires. The matrices $\mathcal{A}_\Omega$ and $\mathcal{A}_{\partial\Omega}$ are possibly singular matrices; $\mathcal{F}_\Omega$ and $\mathcal{F}_{\partial\Omega}$ are functions containing spatial partial differential operators. The matrices $\mathcal{A}_\Omega$ and $\mathcal{A}_{\partial\Omega}$ do not contain space or time derivatives of $u(\underline{x},t)$. The space discretization of (3.1) (method of lines) yields

$$A(t,U)\dot{U} = F(t,U), \quad U(t_0) = U_0, \quad t > t_0, \tag{3.2}$$

where $U(t)$ is the numerical approximation of $u(\underline{x},t)$ on a space grid. If $A(t,U)$ is singular then (3.2) will be a system of differential-algebraic equations (DAEs) and (3.2) will be a system of ordinary differential equations (ODEs), otherwise. In case we have, for example, Neumann or Dirichlet boundary conditions, then $A(t,U)$ will possess rows containing only zeros which implies that (3.2) is a DAE system.

Local uniform grid refinement methods use local subgrids of changing size in time and thus generate solution vectors with a variable dimension. This complicates the analysis. In order to circumvent this problem, the fine local subgrids are expanded over the whole of $\Omega \cup \partial\Omega$. The solution to (3.1) is computed only within the original perimeter of the subgrid and interpolated from the next coarser subgrid outside this region. This is only done in the mathematical formulation of the method to make the error analysis easier. It does *not* take place in the implementation of the method.

Let Ω_k, $1 \le k \le l$, be uniform space grids covering $\Omega \cup \partial\Omega$ with l denoting the maximum number of grid levels needed to advance the solution from t_{n-1} to t_n. The grid refinement is cellular so Ω_k is obtained from Ω_{k-1} by bisecting sides of cells of Ω_{k-1}. Note that nodes of Ω_{k-1} coincide with nodes on Ω_k. Let S_k be the vector space of all grid functions on Ω_k and let $U_k^n \in S_k$ be the approximation to $u(\underline{x},t_n)$ at Ω_k. Suppose that (3.2) is defined on Ω_k, then using an s-step backward differentiation formula (BDF) for time stepping results in the following system of equations,

$$\begin{aligned} \frac{1}{\theta_s \tau} A_k^n(U_k^n)U_k^n &= \frac{1}{\theta_s \tau} A_k^n(U_k^n)V_k^{n-1} + F_k^n(U_k^n), \\ V_k^{n-1} &= a_1 U_k^{n-1} + \cdots + a_s U_k^{n-s}, \quad k=1,....,l, \end{aligned} \tag{3.3}$$

where $\tau = t_n - t_{n-1}$, V_k^{n-1} is the history vector collecting values computed at backward time points and θ_s, a_1, ..., a_s are coefficients depending on current and previous time step sizes.

Our formulation of the grid refinement method uses the following matrix operators:

- *the identity matrix* $I_k : S_k \to S_k$,
- *a diagonal matrix* $D_k^n : S_k \to S_k$, *with diagonal entries equal to unity or zero,* $D_1^n = I_1$,
- *the restriction operator* $R_{lk} : S_l \to S_k$, $R_{ll} = I_l$
- *the interpolation operator* $P_{k-1k} : S_{k-1} \to S_k$.

The matrix D_k^n determines whether the solution at a particular grid node is obtained by solving (3.1) or by interpolation from Ω_{k-1}. The diagonal entries of D_k^n associated with this node are equal to unity when (3.1) is solved and equal to zero otherwise. The number of the diagonal entries associated with each grid node is equal to the number of PDEs. Note that on the coarsest grid, system (3.1) is solved on the whole of Ω_1, meaning that $D_1^n = I_1$. For example, the components of the vector $D_k^n \delta_k^n$, where δ_k^n is an arbitrary vector in S_k, are nonzero when their corresponding nodes lie inside the region where (3.1) is solved, and if interpolation takes place then these components are zero. With the vector $(I_k - D_k^n)\delta_k^n$ it is just the other way around. The injection of fine grid solution values in the coarser grid solution is denoted by the operator R_{lk} and the interpolation by the operator P_{k-1k}. Since all nodes of Ω_k are also contained in Ω_l, injection takes place at each node of Ω_k.

One time step of the grid refinement method consists of l consecutive interpolation and solution steps on grids Ω_k. Those are defined by

$$(I_k - D_k^n)U_k^n = (I_k - D_k^n)\, P_{k-1k}\, U_{k-1}^n, \tag{3.4a}$$

$$\frac{1}{\theta_s \tau} D_k^n A_k^n(U_k^n)U_k^n = \frac{1}{\theta_s \tau} D_k^n A_k^n(U_k^n) R_{lk} V_l^{n-1} + D_k^n F_k^n(U_k^n), \tag{3.4b}$$

$$k=1,\dots,l.$$

Formula (3.4a) represents the interpolation step and (3.4b) the BDF solution step. The subgrids in the local uniform grid refinement method are properly nested. This means that the region of the domain covered by that part of grid level k where (3.1) is solved is covered completely by its counterpart associated with grid level $k-1$. Hence, the set of nodes contained by that part of grid level k where interpolation takes place will also belong to the set of nodes where interpolation takes place at the grid levels $k+1$, $k+2$, ..., l. In other words, the solution at the part of grid level k where interpolation takes place will be repeatedly interpolated until grid level l is reached. Finally, we emphasize that this occurs only in the formulation of the method to make the analysis easier. In practice, interpolation only takes place where it is really needed.

5.4. ERROR ANALYSIS

In this section the results of the error analysis are presented. First, the truncation errors of the interpolation and the space and time discretization will be introduced. Then, using the mathematical formulation (3.4), relations are derived for the local and global error. Finally, we give an example of the behavior of the local and global error when a coupled system of PDEs is solved.

5.4.1. Error relations

For u_k^n, the pointwise restriction to Ω_k of the exact solution $u(\underline{x}, t_n)$ of (3.1), we have the following error relations

$$(I_k - D_k^n)u_k^n = (I_k - D_k^n)\, P_{k-1k}\, u_{k-1}^n + (I_k - D_k^n)\,\gamma_k^n, \tag{4.1a}$$

$$\frac{1}{\theta_s \tau} D_k^n A_k^n(u_k^n)u_k^n = \frac{1}{\theta_s \tau} D_k^n A_k^n(u_k^n) R_{lk} v_l^{n-1} + D_k^n F_k^n(u_k^n) + \tag{4.1b}$$

$$D_k^n(\alpha_k^n - A_k^n(u_k^n)\beta_k^n), \qquad k=1,....,l,$$

where $v_l^{n-1} = a_1 u_l^{n-1} + ... + a_s u_l^{n-s}$ and α_k^n, β_k^n and γ_k^n are the truncation errors of the space discretization, the time discretization and the interpolation, respectively. They are defined by

$$\alpha_k^n = A_k^n(u_k^n)(u_t)_k^n - F_k^n(u_k^n), \tag{4.2a}$$

$$\beta_k^n = (u_t)_k^n - \frac{1}{\theta_s \tau}[u_k^n - a_1 u_k^{n-1} - \cdots - a_s u_k^{n-s}], \tag{4.2b}$$

$$\gamma_k^n = u_k^n - P_{k-1k} u_{k-1}^n, \tag{4.2c}$$

where $(u_t)_k^n$ is the restriction of $u_t(\underline{x}, t_n)$ to Ω_k. Since the restriction operator R_{lk} involves only the replacement of coarse grid values by fine grid values at coinciding nodes after a time step has been performed on all grid levels, no additional errors are introduced here. Therefore, we have $u_k^{n-1} = R_{lk} u_l^{n-1}$. The global error at t_n at Ω_k is defined by

$$e_k^n = u_k^n - U_k^n. \tag{4.3}$$

Subtracting (3.4) from (4.1) we get

$$(I_k - D_k^n)e_k^n = (I_k - D_k^n)P_{k-1k}e_{k-1}^n + (I_k - D_k^n)\gamma_k^n, \tag{4.4a}$$

$$\frac{1}{\theta_s \tau} D_k^n [K_k^n + W_k^n] e_k^n = \frac{1}{\theta_s \tau} D_k^n [L_k^n e_k^n + W_k^n R_{lk} f_l^{n-1}] + \tag{4.4b}$$

$$D_k^n M_k^n e_k^n + D_k^n(\alpha_k^n - A_k^n(u_k^n)\beta_k^n),$$

$$k=1,....,l,$$

where

$$f_l^{n-1} = a_1 e_l^{n-1} + \cdots + a_s e_l^{n-s}, \tag{4.5}$$

and

$$[K_k^n + W_k^n] e_k^n = \int_0^1 \frac{d}{d\theta}[A_k^n(\xi)\xi] d\theta = A_k^n(u_k^n)u_k^n - A_k^n(U_k^n)U_k^n, \tag{4.6a}$$

$$L_k^n e_k^n + W_k^n R_{lk} f_l^{n-1} = \int_0^1 \frac{d}{d\theta}[A_k^n(\xi)\eta]d\theta \tag{4.6b}$$

$$= A_k^n(u_k^n)R_{lk}v_l^{n-1} - A_k^n(U_k^n)R_{lk}V_l^{n-1},$$

$$M_k^n e_k^n = \int_0^1 \frac{d}{d\theta}[F_k^n(\xi)]d\theta = F_k^n(u_k^n) - F_k^n(U_k^n), \tag{4.6c}$$

$$W_k^n = \int_0^1 A_k^n(\xi)d\theta, \quad K_k^n = \int_0^1 \frac{d}{d\tilde{\xi}}[A_k^n(\tilde{\xi})\xi]d\theta, \quad L_k^n = \int_0^1 \frac{d}{d\tilde{\xi}}[A_k^n(\tilde{\xi})\eta]d\theta, \tag{4.6d}$$

$$\xi = \tilde{\xi} = \theta u_k^n + (1-\theta)U_k^n, \quad \eta = \theta R_{lk}v_l^{n-1} + (1-\theta)R_{lk}V_l^{n-1}, \tag{4.6e}$$

which are obtained by applying the mean value theorem for vector functions. Combining (4.4a) and (4.4b) yields a recurrence relation for the global error

$$e_k^n = \Gamma_k^n f_l^{n-1} + X_k^n e_{k-1}^n + \phi_k^n, \quad k=1,\dots,l, \tag{4.7}$$

where

$$\Gamma_k^n = (Z_k^n)^{-1}\frac{1}{\theta_s\tau}D_k^n W_k^n R_{lk} \tag{4.8a}$$

$$X_k^n = (Z_k^n)^{-1}(I_k - D_k^n)P_{k-1k}, \tag{4.8b}$$

$$\phi_k^n = (Z_k^n)^{-1}\{D_k^n\alpha_k^n - D_k^n A_k^n(u_k^n)\beta_k^n + (I_k - D_k^n)\gamma_k^n\}, \tag{4.8c}$$

$$Z_k^n = I_k - D_k^n + D_k^n(\frac{1}{\theta_s\tau}[W_k^n + K_k^n - L_k^n] - M_k^n). \tag{4.8d}$$

The vector ϕ_k^n is the local level error which is the contribution associated with a single time step of grid level k to the global error e_k^n and Z_k^n is the integrated Jacobian of the system of equations. Using the specific form of Z_k^n, we note that Z_k^n can be written as $I_k - D_k^n + D_k^n Z_k^n$. When $(Z_k^n)^{-1}$ is written as $(I_k - D_k^n)(Z_k^n)^{-1} + D_k^n(Z_k^n)^{-1}$, it can by pre-multiplying $(Z_k^n)^{-1}$ with Z_k^n very easily be shown that

$$(Z_k^n)^{-1} = I_k - D_k^n + D_k^n(Z_k^n)^{-1}, \quad k=1,\dots,l. \tag{4.9}$$

Relation (4.7) is similar to the one obtained in [8-11] and leads to the following expressions for the global and local error.

$$e_k^n = G_k^n f_l^{n-1} + \psi_k^n, \quad n=1,2,\dots; \quad k=1,\dots,l, \tag{4.10}$$

where G_k^n is the amplification matrix and ψ_k^n the local error which is the contribution associated with one time step of k grid levels to the global error e_k^n. They are given by

$$G_1^n = \Gamma_1^n, \qquad G_k^n = X_k^n \, G_{k-1}^n + \Gamma_k^n, \tag{4.11a}$$

$$\psi_1^n = \phi_1^n, \qquad \psi_k^n = X_k^n \, \psi_{k-1}^n + \phi_k^n, \qquad k=2,....,l, \tag{4.11b}$$

which result in

$$G_k^n = \sum_{j=1}^{k-1} (\prod_{i=k}^{j+1} X_i^n) \, \Gamma_j^n + \Gamma_k^n, \tag{4.12a}$$

$$\psi_k^n = \sum_{j=1}^{k-1} (\prod_{i=k}^{j+1} X_i^n) \, \phi_j^n + \phi_k^n, \qquad k=2,....,l. \tag{4.12b}$$

Now the local level error ϕ_k^n (4.8c) can be split up in a spatial part $\phi_{k,s}^n$ and a temporal part $\phi_{k,t}^n$, where $\phi_k^n = \phi_{k,s}^n + \phi_{k,t}^n$. These parts are given by

$$\phi_{k,s}^n = (Z_k^n)^{-1} [D_k^n \alpha_k^n \; + \; (I_k - D_k^n) \gamma_k^n], \tag{4.13a}$$

$$\phi_{k,t}^n = \; - (Z_k^n)^{-1} D_k^n A_k^n (u_k^n) \beta_k^n. \tag{4.13b}$$

This yields two distinct relations for the local space error $\psi_{k,s}^n$ and the local time error $\psi_{k,t}^n$. With (4.9), (4.11b) and (4.13a), the relations for $\psi_{k,s}^n$ read

$$D_k^n \psi_{k,s}^n \; = \; D_k^n (Z_k^n)^{-1} \{ D_k^n \alpha_k^n \; + \; (I_k - D_k^n) [P_{k-1k} \psi_{k-1,s}^n + \gamma_k^n] \}, \tag{4.14a}$$

$$(I_k - D_k^n) \psi_{k,s}^n \; = \; (I_k - D_k^n) [P_{k-1k} \psi_{k-1,s}^n + \gamma_k^n]. \tag{4.14b}$$

Here $D_k^n \psi_{k,s}^n$ denotes the local space error inside the region of grid level k where (3.1) is solved and $(I_k - D_k^n) \psi_{k,s}^n$ the local space error outside this region. In a similar way, the local time error can be written as

$$D_k^n \psi_{k,t}^n \; = \; D_k^n (Z_k^n)^{-1} \{ \; - D_k^n A_k^n (u_k^n) \beta_k^n \; + \; (I_k - D_k^n) P_{k-1k} \psi_{k-1,t}^n \}, \tag{4.15a}$$

$$(I_k - D_k^n) \psi_{k,t}^n \; = \; (I_k - D_k^n) P_{k-1k} \psi_{k-1,t}^n. \tag{4.15b}$$

Since the local error can be split up in a spatial and temporal part, the same can be done with the global error. Using (4.7)-(4.9) and (4.13) we get similar relations for the global space error

$$D_k^n e_{k,s}^n = D_k^n (Z_k^n)^{-1} \{ D_k^n [\frac{1}{\theta_s \tau} W_k^n R_{lk} f_{l,s}^{n-1} + \alpha_k^n] + \tag{4.16a}$$

$$(I_k - D_k^n)[P_{k-1k} e_{k-1,s}^n + \gamma_k^n] \},$$

$$(I_k - D_k^n) e_{k,s}^n = (I_k - D_k^n)[P_{k-1k} e_{k-1,s}^n + \gamma_k^n] \}, \tag{4.16b}$$

and for the global time error

$$D_k^n e_{k,t}^n = D_k^n (Z_k^n)^{-1} \{ D_k^n [\frac{1}{\theta_s \tau} W_k^n R_{lk} f_{l,t}^{n-1} - A_k^n(u_k^n) \beta_k^n] + \tag{4.17a}$$

$$(I_k - D_k^n) P_{k-1k} e_{k-1,t}^n \},$$

$$(I_k - D_k^n) e_{k,t}^n = (I_k - D_k^n) P_{k-1k} e_{k-1,t}^n . \tag{4.17b}$$

When we consider for example the global space error, given by (4.16), we observe that the global space error in the region of grid level k where interpolation takes place (4.16b) is determined by the global space error at grid level $k-1$ and the interpolation error. The global space error in the region of grid level k where (3.1) is solved (4.16a) is determined by the inverse of the Jacobian Z_k^n operating on a vector which consists of a part due to the spatial discretization error and the global space error at previous time points, only nonzero inside this region, and a part due to the global space error outside this region.

5.4.2. Error behavior

In this paragraph an example will be given of the behavior of the global and local error. We consider a system of two coupled PDEs which is solved on a single grid. Therefore, we will drop the subscripts denoting grid levels, k and l, in the remainder of this section. This system of PDEs leads to the following system of differential-algebraic equations after spatial discretization

$$\begin{pmatrix} I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \dot{U}_1 \\ \dot{U}_2 \end{pmatrix} = \begin{pmatrix} F_1(t, U_1, U_2) \\ F_2(t, U_1, U_2) \end{pmatrix}, \tag{4.18}$$

which is written in the format (3.2). The BDF method (3.3) is applied to solve (4.18) and some notation will be introduced needed for the examination of the global and local errors. Relations for the local and global error are derived, using (4.7) and (4.8). Due to (4.6c), (4.6d) and (4.18), we have

$$W^n = \begin{pmatrix} I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad K^n = L^n = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad M^n = \begin{pmatrix} M_{11}^n & M_{12}^n \\ M_{21}^n & M_{22}^n \end{pmatrix}. \tag{4.19}$$

The matrix D_k^n appearing in the error relation (4.7) will be equal to the identity matrix I_k in this case which implies that the $(I_k - D_k^n)$-terms vanish from (4.7) and (4.8). The matrix Z^n is now given by

$$Z^n = \begin{bmatrix} \frac{1}{\theta_s \tau} I - M_{11}^n & -M_{12}^n \\ -M_{21}^n & -M_{22}^n \end{bmatrix} \tag{4.20}$$

Further, the matrix R_{lk} will be equal to the identity matrix. Using the notation above, this relation (4.7) for the global error is now given by

$$\begin{bmatrix} e_1^n \\ e_2^n \end{bmatrix} = \begin{bmatrix} \frac{1}{\theta_s \tau} I - M_{11}^n & -M_{12}^n \\ -M_{21}^n & -M_{22}^n \end{bmatrix}^{-1} * \tag{4.21}$$

$$\left\{ \frac{1}{\theta_s \tau} \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} f_1^{n-1} \\ f_2^{n-1} \end{bmatrix} + \begin{bmatrix} \alpha_1^n \\ \alpha_2^n \end{bmatrix} - \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \beta_1^n \\ \beta_2^n \end{bmatrix} \right\}.$$

The global error at the current time t_n, e^n is given by $(e_1^n, e_2^n)^T$ and the vector f^{n-1} in which global errors at backward times are collected by $(f_1^{n-1}, f_2^{n-1})^T$ (cf. (4.5)). The vectors e_1^n and e_2^n represent the global error belonging to U_1^n and U_2^n respectively. Now examining (4.21) leads to the conclusion that only f_1^{n-1} contributes to e^n. This means that in this case only e_1^n carries over to future time points while e_2^n does not.

For the local error ψ^n we have, according to (4.21)

$$\begin{bmatrix} \psi_1^n \\ \psi_2^n \end{bmatrix} = \begin{bmatrix} \frac{1}{\theta_s \tau} I - M_{11}^n & -M_{12}^n \\ -M_{21}^n & -M_{22}^n \end{bmatrix}^{-1} \left\{ \begin{bmatrix} \alpha_1^n \\ \alpha_2^n \end{bmatrix} - \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \beta_1^n \\ \beta_2^n \end{bmatrix} \right\}. \tag{4.22}$$

The behavior of the local error for small τ is determined by the operator $(Z^n)^{-1}$. The behavior of this operator when it operates on a vector will be investigated for this case where it is assumed that the cell sizes of the space grid remain constant. This operator can be written as

$$(Z^n)^{-1} = \begin{bmatrix} \frac{1}{\theta_s \tau} I - M_{11}^n & -M_{12}^n \\ -M_{21}^n & -M_{22}^n \end{bmatrix}^{-1} = \begin{bmatrix} Y_{11}^n & Y_{12}^n \\ Y_{21}^n & Y_{22}^n \end{bmatrix}. \tag{4.23}$$

When we assume that the diagonal blocks of Z^n are nonsingular we obtain

$$\begin{aligned}
Y_{11}^n &= [I + (\frac{1}{\theta_s \tau} I - M_{11}^n)^{-1} M_{12}^n (M_{22}^n)^{-1} M_{21}^n]^{-1} * \qquad (4.24a)\\
&\quad (\frac{1}{\theta_s \tau} I - M_{11}^n)^{-1},\\
Y_{21}^n &= -(M_{22}^n)^{-1} M_{21}^n Y_{11}^n, \qquad (4.24b)\\
Y_{22}^n &= -[I + (M_{22}^n)^{-1} M_{21}^n (\frac{1}{\theta_s \tau} I - M_{11}^n)^{-1} M_{12}^n]^{-1} (M_{22}^n)^{-1} \qquad (4.24c)\\
Y_{12}^n &= (\frac{1}{\theta_s \tau} I - M_{11}^n)^{-1} M_{12}^n Y_{22}^n. \qquad (4.24d)
\end{aligned}$$

This leads to the following approximations for small τ

$$\begin{aligned}
&Y_{11}^n \approx \theta_s \tau I, \quad Y_{21}^n \approx -\theta_s \tau (M_{22}^n)^{-1} M_{21}^n, \qquad (4.25)\\
&Y_{22}^n \approx -(M_{22}^n)^{-1}, \quad Y_{12}^n \approx -\theta_s \tau M_{12}^n (M_{22}^n)^{-1},
\end{aligned}$$

from which we obtain the following approximations for the local error components

$$\begin{aligned}
\psi_1^n &\approx \theta_s \tau I(\alpha_1^n - \beta_1^n) - \theta_s \tau M_{12}^n (M_{22}^n)^{-1} \alpha_2^n, \qquad (4.26a)\\
\psi_2^n &\approx -\theta_s \tau (M_{22}^n)^{-1} M_{21}^n (\alpha_1^n - \beta_1^n) - (M_{22}^n)^{-1} \alpha_2^n. \qquad (4.26b)
\end{aligned}$$

We see from (4.26) that the first component of the local error ψ_1^n consists of the truncation errors of the space and time discretization multiplied with an operator which behaves like $O(\tau)$ for $\tau \rightarrow 0$, meaning that this component of the local error will vanish in this case. Further we see that, unlike ψ_1^n, the component ψ_2^n does not disappear completely when $\tau \rightarrow 0$.

From this example we conclude that when a system of coupled PDEs are solved the local and global error components can exhibit a very different behavior. Therefore, a refinement strategy of an adaptive grid method based on error estimation will have to take such differences in behavior into account. This means that in case of a system of coupled PDEs, distinction must be made between errors associated with each separate PDE when we want to develop a refinement strategy based on error estimations. This will be discussed in the next section.

In [8] PDEs were considered which upon discretization of the space derivatives lead to a system of explicit ODEs in which the boundary conditions were incorporated. This system of ODEs was solved using the implicit Euler method. The idea was to control the spatial accuracy of the solution by controlling the local space error. Moreover, in case the number of grid levels is constant in time, the local space error at the finest local subgrid should be comparable to the local space error on a single, uniform grid having the same cell sizes as the finest subgrid. A refinement strategy was developed aiming to fulfill the following inequality which is called *the refinement condition*

$$\|(Z_l^n)^{-1}(I_l - D_l^n)[P_{l-1l}\psi_{l-1,s}^n + \gamma_l^n]\|_\infty \; < \; c\|(Z_l^n)^{-1}\tau D_l^n \alpha_l^n\|_\infty, \qquad c > 0, \tag{5.1}$$

where Z_l^n is defined as

$$Z_l^n \; = \; I_l - \tau D_l^n M_l^n . \tag{5.2}$$

The matrices I_l and D_l^n are defined similarly as in Section 3. The matrix M_l^n arises after discretizing the space derivatives of a time-dependent PDE and is comparable to the matrix defined by (4.6c). When $\|(Z_l^n)^{-1}\|_\infty \le 1$, (5.1) results in the following bound for the local space error at the finest subgrid

$$\|\psi_{l,s}^n\|_\infty \; < \; (1{+}c)\|(Z_l^n)^{-1}\tau D_l^n \alpha_l^n\|_\infty . \tag{5.3}$$

Apart from the constant c, this error bound is similar to the error bound we would get using a single, uniform grid. This indicates that by satisfying the refinement condition it is possible to get more or less the same spatial accuracy as if a single, uniform grid was used. Further, it was proved that (5.1) holds when the inequality

$$\max_{2 \le j \le l} \{\|(I_j - D_j^n)\lambda_j^n\|_\infty\} \le \frac{c}{l-1}\|(Z_l^n)^{-1}\tau D_l^n \alpha_l^n\|_\infty , \tag{5.4}$$

is satisfied while creating finer, local, uniform subgrids, where

$$\lambda_j^n = \gamma_j^n + P_{j-1j}(Z_{j-1}^n)^{-1}\tau D_{j-1}^n \alpha_{j-1}^n . \tag{5.5}$$

In [9] a similar refinement condition and error bound were derived based on a general Runge-Kutta time stepping scheme and in [11] a refinement condition was derived for elliptic PDEs.

Although the refinement strategy based on fulfilling (5.4) worked satisfactorily in practice, it stems from rather conservative estimates of norms and can therefore be restrictive, especially when the number of grid levels is large. Further, in order to fulfill (5.4) when $\tau \rightarrow 0$ then the matrix D_k^n inevitably has to approach the identity matrix I_k. This means that when τ is decreased, the local subgrids will cover an increasingly larger part of the domain and will in the limit of $\tau = 0$ cover the entire domain. In this respect, (5.4) is also restrictive and one can argue whether it is really necessary to let the local subgrids cover a larger part of the domain with decreasing τ in order to retain a high spatial accuracy. Finally, from the previous section we have concluded that components of the global and local error can exhibit a different behavior. This is due to the fact that these different components are associated with different PDEs. The inequalities (5.1) and (5.4) and the error bound (5.3) are based on estimates of matrix norms where it is assumed that the system of PDEs at hand lead to a system of explicit ODEs after spatial discretization. Hence, it is assumed that all local and global error components behave in a similar manner. This implies that (5.1) and (5.4) are not sufficiently accurate in case a coupled system of general PDEs is solved. This might also be the case with a system of coupled PDEs where upon semi discretization one or more PDEs lead to a much stiffer system of ODEs than the other ones. For this reason, a new, more general refinement strategy is developed for a general system of coupled PDEs. In contrast to most of our previous work, this new strategy will be based on controlling the global space error which is less restrictive than (5.1). It should satisfy two demands. First, it must make a distinction between vector components associated with different PDEs, and second, it must computationally not be too expensive.

Basically a refinement strategy has to answer two questions. The first one is, *when should a new finer grid level be created*, and the second one is, *which grid cells need to be refined*. In order to answer these questions we will now introduce some notation. Suppose (3.1) consists of q different PDEs in which the boundary conditions are included, i.e. boundary conditions are regarded as separate PDEs defined on boundaries only. Note that a single PDE with Neumann or Dirichlet boundary conditions can also be regarded this way.

An arbitrary vector $\delta_k^n \in S_k$ is generically denoted as $(\delta_{k,1}^n, \delta_{k,2}^n, \cdots, \delta_{k,q}^n)^T$, where the component $\delta_{k,j}^n$ is associated with the j^{th} PDE. The matrices I_k, D_k^n, P_{k-1k}, R_{lk} are block diagonal and can be written as

$$I_k = \operatorname{diag}(I_{k,1},\ I_{k,2},\ \ldots,\ I_{k,q}), \tag{5.6a}$$

$$D_k^n = \operatorname{diag}(D_{k,1}^n,\ D_{k,2}^n,\ \ldots,\ D_{k,q}^n), \tag{5.6b}$$

$$P_{k-1k} = \operatorname{diag}(P_{k-1k,1},\ P_{k-1k,2},\ \ldots,\ P_{k-1k,q}), \tag{5.6c}$$

$$R_{lk} = \operatorname{diag}(R_{lk,1},\ R_{lk,2},\ \ldots,\ R_{lk,q}). \tag{5.6d}$$

The matrices W_k^n, K_k^n, L_k^n, M_k^n and Z_k^n are written as block matrices with the blocks $W_{k,ij}^n$, $K_{k,ij}^n$, $L_{k,ij}^n$, $M_{k,ij}^n$ and $Z_{k,ij}^n$, where $i,j = 1,\ldots.,q$. The blocks of Z_k^n are given by

$$Z^n_{k,ii} \;=\; I_{k,i} - D^n_{k,i} \;+\; D^n_{k,i}(\frac{1}{\theta_s \tau}[W^n_{k,ii} + K^n_{k,ii} - L^n_{k,ii}] \;-\; M^n_{k,ii}), \qquad (5.7)$$

$$Z^n_{k,ij} \;=\; D^n_{k,i}(\frac{1}{\theta_s \tau}[W^n_{k,ij} + K^n_{k,ij} - L^n_{k,ij}] \;-\; M^n_{k,ij}), \qquad i \neq j.$$

This leads to the following relation for the global space error

$$e^n_{k,s,i} \;=\; \sum_{j=1}^{q} Y^n_{k,ij} \{ D^n_{k,j} [\frac{1}{\theta_s \tau} \sum_{m=1}^{q} W^n_{k,jm} R_{lk,m} f^{n-1}_{l,s,m} \;+\; \alpha^n_{k,j}] \;+ \qquad (5.8)$$

$$(I_{k,j} - D^n_{k,j})[P_{k-1k,j} e^n_{k-1,s,j} \;+\; \gamma^n_{k,j}]\},$$

$$i=1,....,q \;; \quad k=1,....,l,$$

where $Y^n_{k,ij}$ is a block of $(Z^n_k)^{-1}$. We have established that the various components of the global and local space error can behave very differently. Consequently, a criterion like, *refine the grid when the local space error ψ^n_k or the global space error e^n_k exceeds some tolerance*, is simply too crude. Although the operators $(Z^n_k)^{-1}$ and W^n_k determine how the truncation error of the space discretization α^n_k affects the spatial accuracy, one can say that the source of the space error is α^n_k. This is certainly true when a single, uniform grid is used. Therefore, the following criterion can be used. A new grid level $k+1$ is created if there exists a component i for which

$$\|\alpha^n_{k,i}\|_\infty \;\geq\; TOL, \qquad i=1,....,q, \qquad (5.9)$$

holds, where TOL is a user-defined tolerance. It is assumed here that the PDE problem at hand is properly scaled. Otherwise the scaling of the various PDEs have to be taken into account in criterion (5.9). We have also built in an extra condition in our research code to smoothen its behavior. Suppose that the maximum number of grid levels during the previous time step is l and that at grid level $k<l$, $\|\alpha^n_{k,i}\|_\infty < TOL$ for all i. Although this means that a new finer grid level $k+1$ is actually not necessary, it will still be created when $\|\alpha^n_{k,i}\|_\infty > 0.9 \times TOL$, to avoid fluctuations of the maximum number of grid levels from one time point to the next.

Further, it should be pointed out that when the number of grid levels is increasing in time additional interpolation errors are introduced, because new initial values have to be interpolated over the whole newly created grid. It is possible that these extra interpolation errors diminish the spatial accuracy. For example, this can be the case when solving reaction diffusion problems with a small diffusion coefficient. In such case, the interpolation error which will be committed when an extra grid level is introduced can be very large due to a steep solution while the global space error is small. For this reason it may be necessary to use an extra criterion to create a new finer grid level based on controlling this potential interpolation error. This means that the situation can occur that grid refinement takes place not to reduce the global space error but to reduce the potential interpolation error. However, in most cases when these potential interpolation errors are large, the global space error will also be large and therefore, a criterion like (5.9) will be sufficient to control the global spatial accuracy. In the refinement strategy we have implemented, only (5.9) is used as a refinement criterion.

Having devised a criterion to generate new finer grid levels, we now have to find a criterion to determine which grid cells need to be refined. In order to do this we use (4.9) to rewrite (5.8) as

$$D^n_{k,i} e^n_{k,s,i} = D^n_{k,i} \sum_{j=1}^{q} Y^n_{k,ij} \{ D^n_{k,j} [\frac{1}{\theta_s \tau} \sum_{m=1}^{q} W^n_{k,jm} R_{lk,m} f^{n-1}_{l,s,m} + \alpha^n_{k,j}] + \quad (5.10a)$$

$$(I_{k,j} - D^n_{k,j}) [P_{k-1k,j} e^n_{k-1,s,j} + \gamma^n_{k,j}] \},$$

$$(I_{k,i} - D^n_{k,i}) e^n_{k,s,i} = (I_{k,j} - D^n_{k,j}) [P_{k-1k,j} e^n_{k-1,s,j} + \gamma^n_{k,j}] \quad (5.10b)$$

$$i=1,....,q \; ; \quad k=1,....,l.$$

When we for a moment abandon the idea of expanded grids and think within terms of local subgrids, then this new criterion will be based on the notion that after the coarser grid values have been replaced by the finer grid values at coinciding nodes, the largest absolute nodal value of the global space error should be at the finest grid level. If this is not the case then the maximum norm of the global space error over all grids will not be reduced by creating the finest grid level, which means that this finest grid level is of no use.

When a grid is locally refined, the nodal values at the part of the grid which is refined are eventually replaced by finer grid values. However, the nodal values outside this part of the domain remain unchanged. This means that if the nodal value of the maximum space error should be at the finest grid level, we have to make sure that the value of the global space error at nodes outside the part of a grid which is going to be refined are smaller than the maximum global space error at the finest grid level. This means that, returning to the expanded grids, the global space error at the part of a grid level where interpolation takes place is smaller than the maximum global space error in the region of the finest grid level where (3.1) is solved. In other words, we have to demand that

$$\| (I_{k,i} - D^n_{k,i}) [P_{k-1k,i} e^n_{k-1,s,i} + \gamma^n_{k,i}] \|_\infty \le c \| D^n_{l,i} e_{l,s,i} \|_\infty , \quad (5.11)$$

$$i=1,....,q \; ; \quad k=2,....,l \; ; \quad 0 < c \le 1,$$

where c is a user-defined constant and l the finest grid level which is going to be used during this time step. If at grid level $k-1$ (5.9) holds for PDE component i and if at node j the associated component of the vector $(I_{k,i} - D^n_{k,i})e^n_{k,s,i}$ satisfies

$$|((I_{k,i} - D^n_{k,i})[P_{k-1k,i}e^n_{k-1,s,i} + \gamma^n_{k,i}])_j| > c\|D^n_{l,i}e_{l,s,i}\|_\infty, \tag{5.12}$$

then the sixteen cells surrounding j will be refined. This implies that the refinement is only controlled by the error components connected with the space discretization error components for which (5.9) holds. In practice the right hand side of (5.12) is estimated at grid level $k-1$. Not all grid level-$k-1$ nodes are scanned but only the nodes within the region of $k-1$ where the PDEs are solved. The reason for this is that outside the region where the PDEs are solved only repeated interpolation takes place until grid level l is reached. The interpolation error committed by repeated interpolation will be bounded since repeated interpolation implies that only more intermediate points will be computed on the same interpolation polynomial. Hence, the estimate of the right hand side of (5.12) at $k-2$ can be regarded as a first-order approximation to the estimate at $k-1$ for the nodes lying outside the region of $k-1$ where the PDEs are solved. This means that when in the region of $k-1$ where interpolation takes place, (5.12) did not hold at a node belonging to $k-2$, we can assume that it will not hold at its corresponding node plus its nearest neighbors at $k-1$ either.

We use (4.16) to compute the global space error which implies that on top of solving (3.4) for the solution we solve an extra equation for the global space error. The spatial discretization error α^n_k is estimated by computing $F^n_k(U^n_k)$ in (3.3) with a higher- and a lower-order discretization and subtracting the two. The interpolation error γ^n_k is computed by numerically estimating the truncation error. For both estimates the numerically computed solution is used. To estimate the right hand side of (5.11) we use the asymptotic behavior of the global space error. In case the space discretization is of order p we have

$$\|D^n_{l,i}e_{l,s,i}\|_\infty \approx 2^{-p(l-k+1)}\|D^n_{k-1,i}e_{k-1,s,i}\|_\infty. \tag{5.13}$$

If $\|\alpha^n_{k-1,i}\|_\infty$ is computed and (5.9) holds then using the asymptotics we can estimate how many grid levels are needed to achieve that (5.9) does not hold any longer. The maximum number of grid levels l which are necessary during this time step is then estimated as

$$l = k + \text{entier}\left\{\frac{\log(\|\alpha^n_{k-1,i}\|_\infty) - \log(TOL)}{p\log(2)}\right\}, \quad i=1,....,q. \tag{5.14}$$

This means that for component i we need l grid levels in order to fulfill $\|\alpha^n_{l,i}\|_\infty < TOL$. This l value is used in (5.13) to estimate $\|D^n_{l,i}e_{l,s,i}\|_\infty$. Note that for different PDE components we can have different l values. The estimates above might not always be accurate, especially at times when the number of grid levels in use is about to change. This may lead to a refinement criterion (5.12) which is too restrictive or not restrictive enough. Only the latter can influence the accuracy in a negative manner. However, chosing c in (5.14) sufficiently small can overcome this problem. It is also possible to improve the estimates of $||D^n_{l,i}e_{l,s,i}||_\infty$ and l by taking these values at the previous time step into consideration.

Finally, we conclude this section with the following remarks. The strategy based on (5.13) does not guarantee that, in case l is fixed over the entire time interval over which the solution is computed, the global space error is comparable to the global space error obtained with a single, uniform grid having the same cell sizes as the finest grid level in the adaptive grid method. If such a guarantee is desired then extra requirements have to be fulfilled in order to get a bound for $||e_{l,s,i}||_\infty$ which is similar to the bound using a single, uniform grid. However, these requirements are difficult to satisfy in practice and they can only be satisfied in an a posteriori manner. For this reason we have not incorporated these requirements in the refinement strategy. Nevertheless, when the constant c decreases then more grid cells are refined and the gap between the spatial error of the adaptive grid and uniform grid computation is likely to become smaller. Further, the strategy described in this section is not the only possible strategy. The relations for the local and global space error, given by (4.14) and (4.16), respectively, leave room for other strategies as well. The new strategy based on (5.11) is less restrictive than the previous one based on (5.4). When the time step size tends to zero, then, according to (5.11), the finer subgrids do not necessarily need to grow. This is in contrast to (5.4) where the subgrids eventually will cover the entire domain. Moreover, the $(l-1)^{-1}$ term is also avoided in (5.11).

5.6. Example problems

Three example problems are used to illustrate the method and to test the refinement strategy. For time integration, Implicit Euler for the first time step and BDF2 with variable coefficients for the following time steps are used. Standard second-order finite differences are used for space discretization and the interpolation is fourth-order Lagrangian.

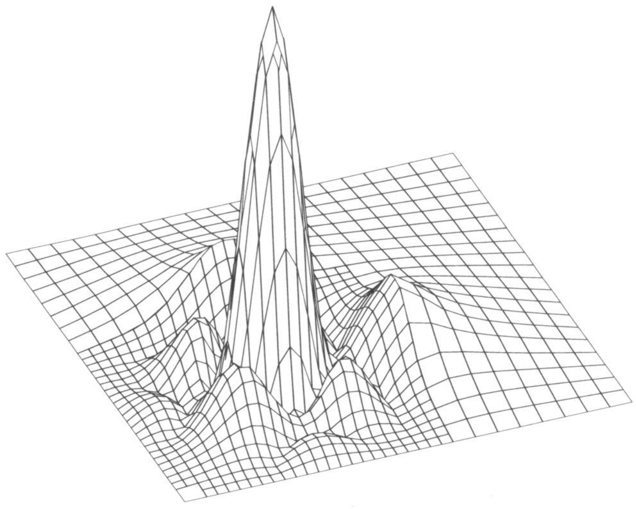
5.6.1. Problem I

This test example is hypothetical and is given by a coupled parabolic and elliptic equation, both linear:

$$u_t = u_{xx} + u_{yy} - v + g(x,y,t), \tag{6.1a}$$

$$0 = v_{xx} + v_{yy} + u + h(x,y,t), \qquad 0<x,y<1, \qquad t>0. \tag{6.1b}$$

The initial function, the Dirichlet boundary conditions and the source terms g and h are selected so that the exact solution is given by



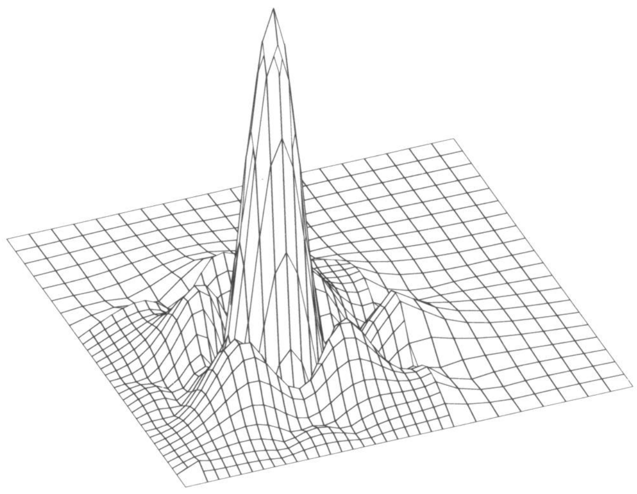
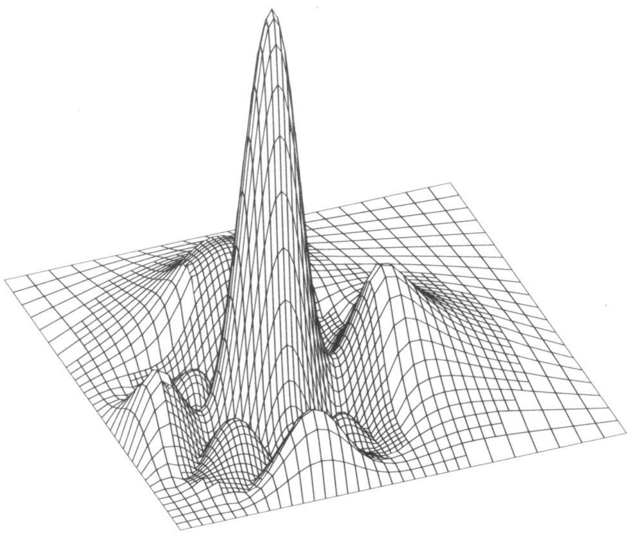


FIGURE 6.1 Problem I. The scaled absolute values of the exact global space error in u and v obtained with 2 grid levels at $t = 0.25$.



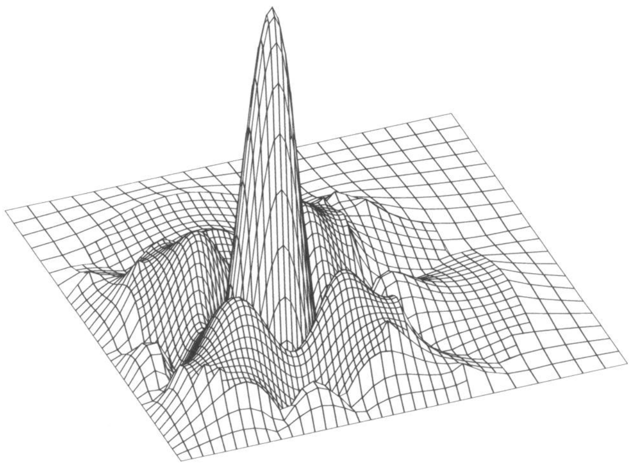
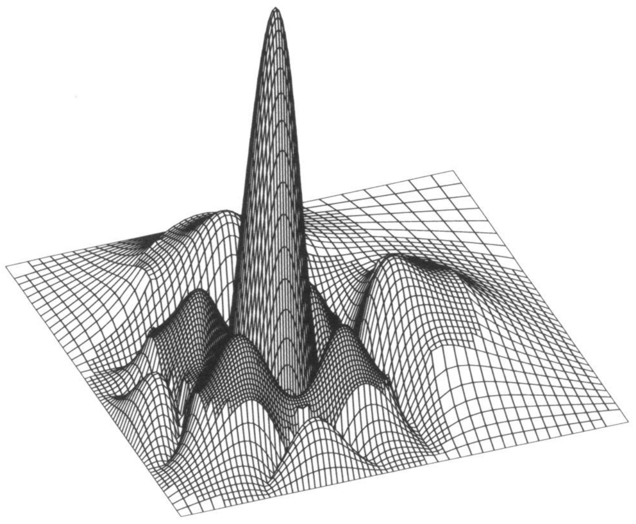


FIGURE 6.1 Continued. The scaled absolute values of the exact global space error in u and v obtained with 3 grid levels at $t = 0.25$.



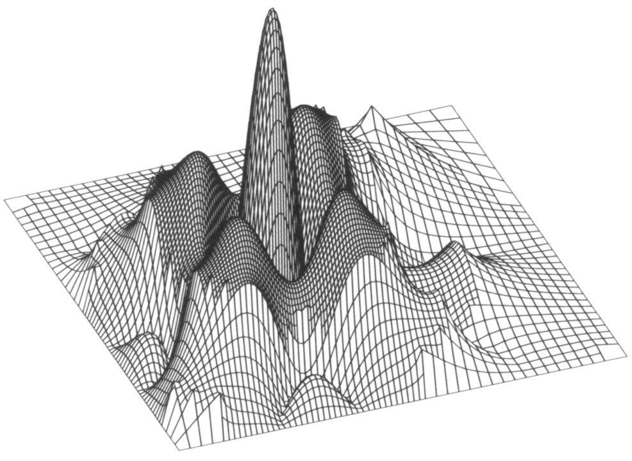
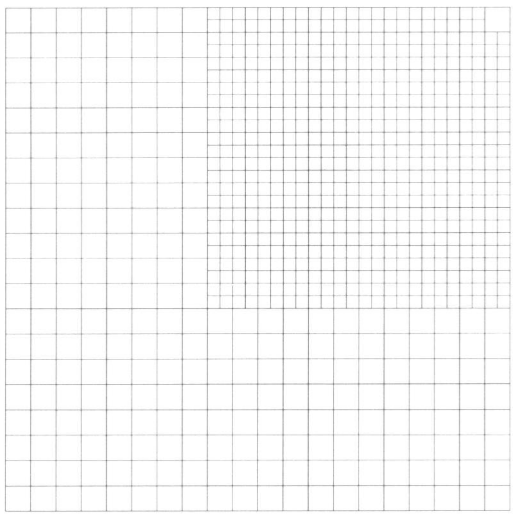


FIGURE 6.1 Continued. The scaled absolute values of the exact global space error in u and v obtained with 4 grid levels at $t = 0.25$.



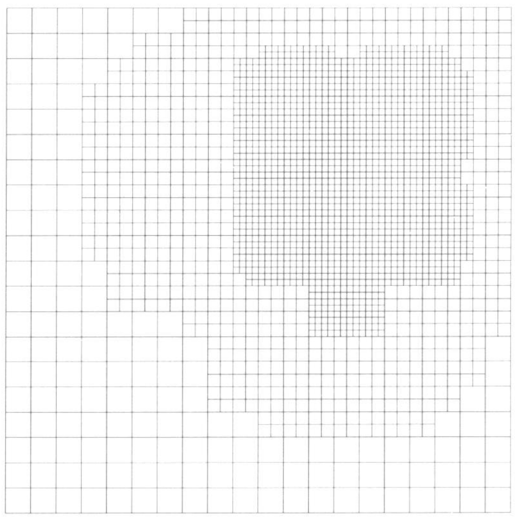


FIGURE 6.2 Problem I. The grids of the 2- and 3-level computation at $t = 0.25$.

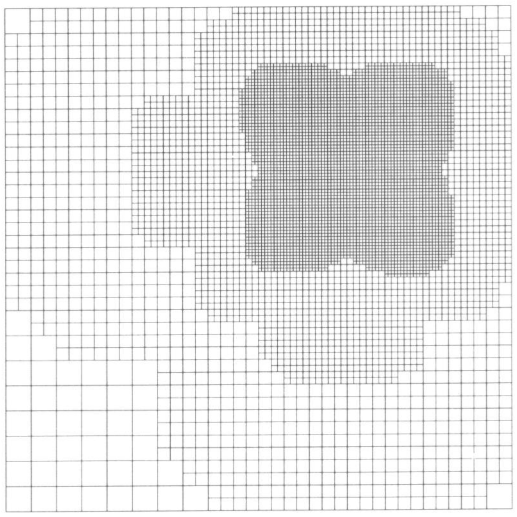


FIGURE 6.2 Continued. The grids of the 4-level computation at $t = 0.25$.

$$u(x,y,t) = v(x,y,t) = \exp[-80((x - r(t))^2 + (y - s(t))^2)], \tag{6.2}$$

where $r(t) = \frac{1}{4}[2 + \sin(\pi t)]$ and $s(t) = \frac{1}{4}[2 + \cos(\pi t)]$. This solution is a cone that is initially centered at $(\frac{1}{2},\frac{3}{4})$ and that rotates around $(\frac{1}{2},\frac{1}{2})$ in a clockwise direction with a constant speed. We have used this problem to subdue the method to a convergence test. The solution was computed from $t = 0$ to $t = 0.25$. Starting from a coarse 20×20 grid, 1,2 and 3 additional grid levels were used. The number of grid levels were kept constant throughout the entire time interval. The associated *TOL* values were 20, 5 and 1. These tolerance values appear to be large compared to the tolerance values one is used to. The reason for this is that the $\|\alpha^n_{k,i}\|_\infty$ values can be large. However the accuracy does not deteriorate severely by this because the inverse of the Jacobian operates on the vector α^n_k (cf. (4.14a), (4.16a)) which reduces the values of the components of this vector considerably. This is due to the large high-frequent components of the grid function α^n_k and the fact that the Jacobian stems from an elliptic/parabolic operator of which the inverse strongly damps such components. The constant time step size was chosen to be equal to 0.005 and the constant c from (5.17) equal to 0.5. The results at the final time, given in Table 6.1, show that the obtained global space errors decrease roughly with a factor of

# of grid levels	global space error u	v
1	0.043973	0.044304
2	0.013465	0.012874
3	0.003594	0.002991
4	0.000757	0.000631

TABLE 6.1 Problem I. Maxima of the exact global space error restricted to the finest grid.

# of grid levels	global space error u	v
1	0.043441	0.044379
2	0.013277	0.013304
3	0.003543	0.003135
4	0.000748	0.000663

TABLE 6.2 Problem I. Maxima of the numerically estimated global space error restricted to the finest grid.

four indicating the normal second-order convergence behavior which would also be obtained with a single, uniform grid. Since the success of the refinement strategy, described in the previous section, depends on the accuracy of error estimates, we have also compared the numerical estimates of the global space error with the exact values. In Table 6.2 the numerical estimates of the global space errors are given and it appears that the estimates are quite accurate. The scaled absolute values of the exact global space error in u and v at $t = 0.25$ for all computations is shown in Figure 6.1. The positioning of the finer subgrids in Figure 6.1 appears to be good. The maximum global space error for both components is located at the finest subgrid in use. Moreover, the refinements are fairly efficient, meaning that not many grid cells are unnecessarily refined. Figure 6.2 shows the grids at $t = 0.25$.

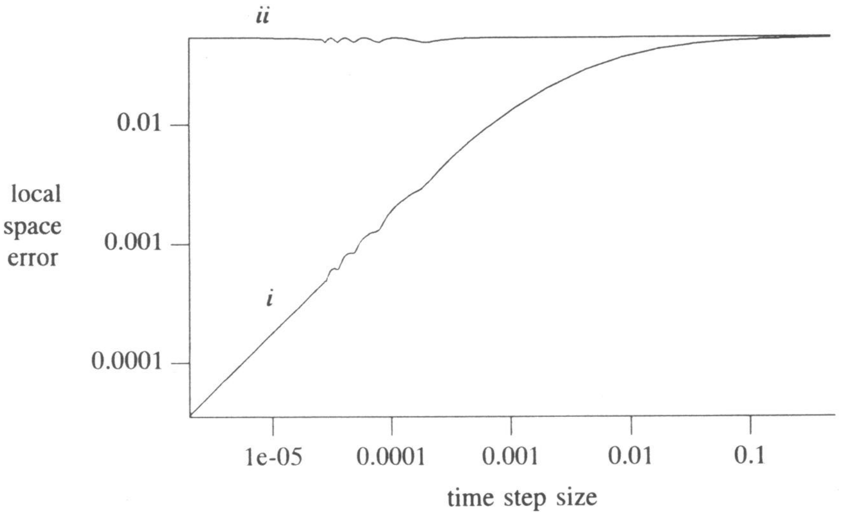


FIGURE 6.3 Problem II. The maximum local space error in u (i) and in v (ii).

5.6.2. Problem II

The second test problem is a problem with a steady-state solution. Again the system of PDEs given by (6.1) is solved but this time the sources g and h and the initial function and Dirichlet boundary conditions are chosen such that the exact solution is given by

$$u(x,y,t) = v(x,y,t) = \exp[-80((x-r(t))^2 + (y-\frac{1}{2})^2)], \tag{6.3}$$

where $r(t)=\frac{1}{2} - \frac{1}{4}\exp(-1000t)$. This represents a cone which is centered at ($\frac{1}{2}$, $\frac{1}{4}$) at t=0 and moves towards the center of the domain ($\frac{1}{2}$, $\frac{1}{2}$) with a continuously decreasing speed. In the steady-state situation the cone will have reached the center of the domain.

Just like in example problem I, (6.1) was solved using 1,2 and 3 extra grid levels after starting from a 20×20 uniform grid. Variable time step sizes were used. On the 20×20 grid at t_n, τ is predicted for the next time step so that $\tau U_t^n = 0.1$, where U_t^n is a numerical approximation of $u_t(\underline{x}, t_n)$. These computed τ values were kept in storage and used as time step sizes for all computations. The constant c from (5.17) is chosen equal to 0.5 for all computations.

This problem was used to illustrate the differences in behavior of the local and

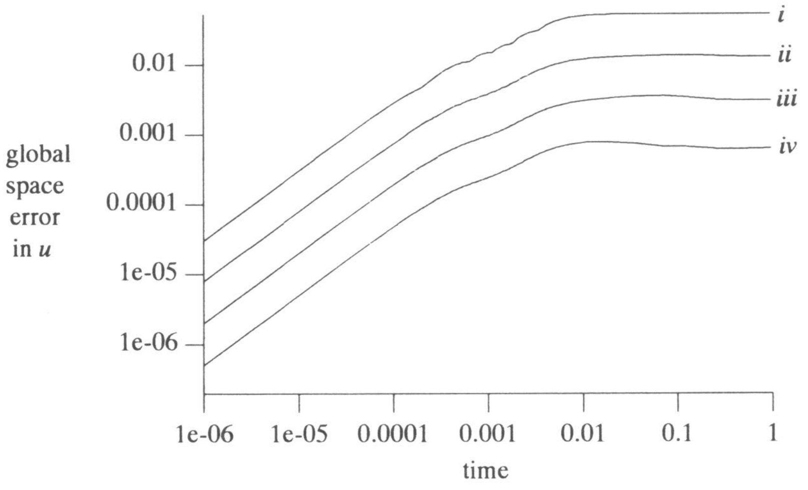


FIGURE 6.4 Problem II. The maximum global space error in u obtained with 1, 2, 3 and 4 grid levels, indicated by *i*, *ii*, *iii* and *iv*, respectively.

global space error belonging to (6.1a) and (6.1b) and to compare the maxima of the global space errors in u and v of the adaptive grid solution obtained with a different number of grid levels throughout the entire time interval. The solution was computed to t=1.0. The *TOL* values were equal to 20, 5 and 1. The number of grid levels were kept constant throughout the entire time interval. The results of paragraph 4.2 apply to this case, since, (6.1) fits in the format (4.18). It is to be expected that the local space error in u behaves like $O(\tau)$ and the local space error in v like $O(1)$ when $\tau \to 0$. Figure 6.3 shows the behavior of the maximum local space error in u and in v over the interior of a 20×20 grid as a function of the time step sizes. A double logarithmic scale was used in this figure and the slope of the local space error in u is almost equal to unity for small time step sizes indicating a linear behavior in τ. Further, the local space error in v appears to be almost constant. This means that the local space error in u and v behave indeed like predicted by (4.26) for small τ. Figure 6.4 and 6.5 compare the maximum global space errors in u and v, respectively, obtained with the adaptive grid method on 1, 2, 3 and 4 grid levels. These figures clearly reveal that the global space error in u, belonging to the PDE (6.1a) gradually increases in time until a certain maximum is reached while the global space error in v, connected with the PDE (6.1b) remains at an almost constant level over the entire time interval. The distances between the lines in Figure 6.4 and 6.5 reveal a second-order convergence behavior which would also be obtained with

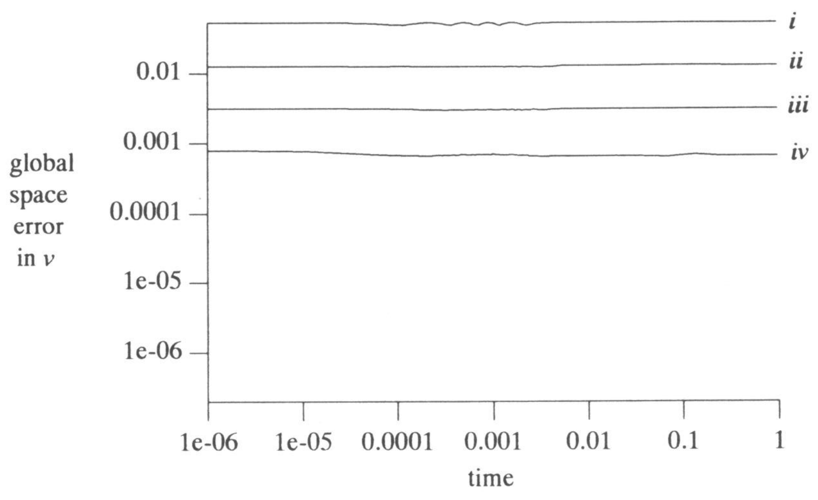


FIGURE 6.5 Problem II. The maximum global space error in v obtained with 1, 2, 3 and 4 grid levels, indicated by *i*, *ii*, *iii* and *iv*, respectively.

a single, uniform grid. The scaled absolute values of the global error in u at t=1.0 are shown in Figure 6.6. The global space error in v is not shown here, because it is very similar to the one in u for this case. Again, the grids are reasonably efficient and the maximum global space error is located at the finest subgrid in use. The grids at the final time are shown in Figure 6.7.

5.6.3. Problem III

The third test problem is a problem with an oscillatory solution. The system of PDEs given by (6.1) is solved once more but this time the sources g and h and the initial function and Dirichlet boundary conditions are chosen such that the exact solution is given by

$$u(x,y,t) = v(x,y,t) = \sin(\pi t)\exp[-320((x-\frac{1}{2})^2 + (y-\frac{1}{2})^2)]. \tag{6.4}$$

This represents an oscillating cone which is centered at (½, ½). At t=0 the solution is zero everywhere. Then a steep pulse emerges at the center of the domain which reaches its maximum at t=0.5 after this it will decay until the solution is equal to zero again at t=1.0.

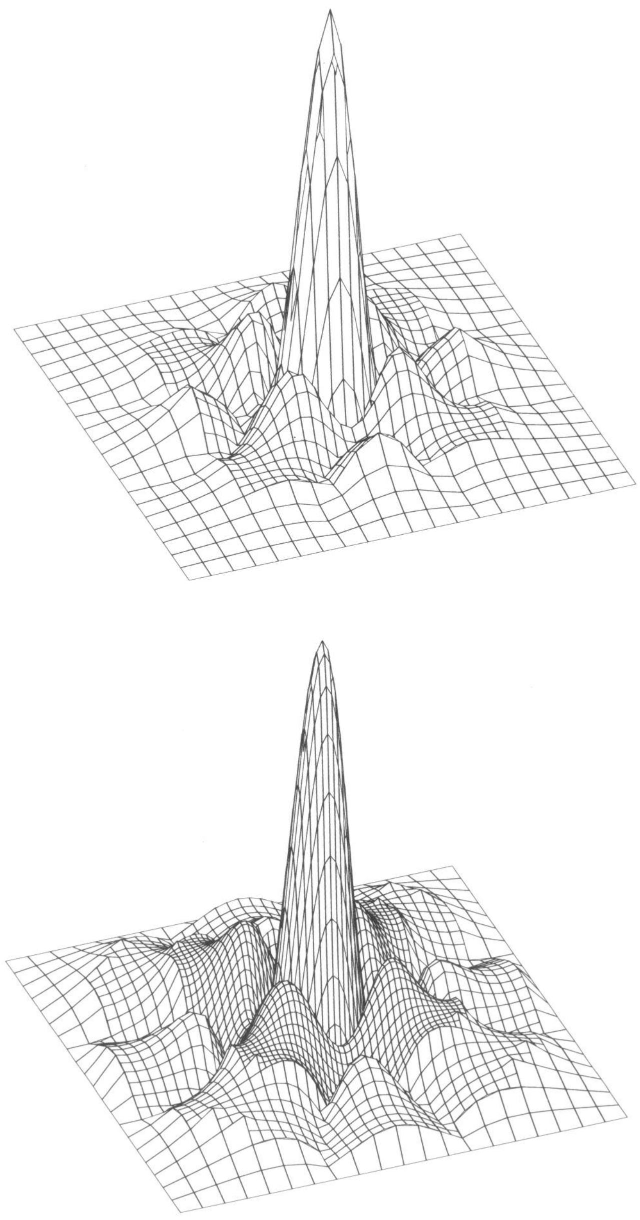


FIGURE 6.6 Problem II. The scaled absolute values of the exact global space error in u obtained with 2 and 3 grid levels at $t = 1.0$.

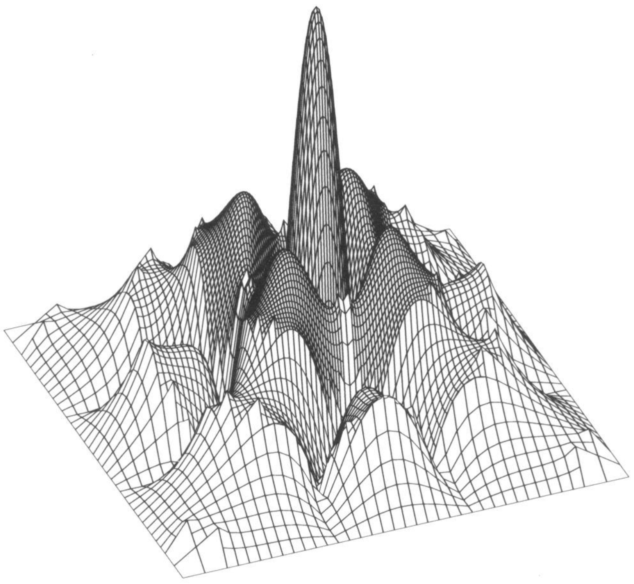


FIGURE 6.6 Continued. The scaled absolute values of the exact global space error in u obtained with 4 grid levels at $t = 1.0$.

This problem was solved to test the performance of the method when a variable number of grid levels is used. The solution was computed four times from t=0 to t=1.0 using a maximum number of grid levels of 2,3,4 and 5. The corresponding *TOL* values were 160,40,10 and 2.5, respectively and the constant c was chosen to be 0.5. Variable time steps were also used here which were determined in exactly the same manner as in problem II and also kept in storage to be used for all computations. The maximum global space error in u as a function of time is shown in Figure 6.8. Here, the behavior of the global space error in v is very similar to the one in u. The kinks in this figure indicate that at that time, a new finer grid level is created or discarded. It appears that the global space error decreases with a certain factor when the *TOL* value is divided by four. Inspection of the data revealed that this factor is larger than four in both the infinity and the L_1 norm. These norms of the maximum global space error were taken over the values at all time levels. This implies that when the *TOL* value is decreased by a factor of four that the spatial accuracy is increased by a factor of at least four for this example problem. The grids at t=0.5 are shown in Figure 6.9.

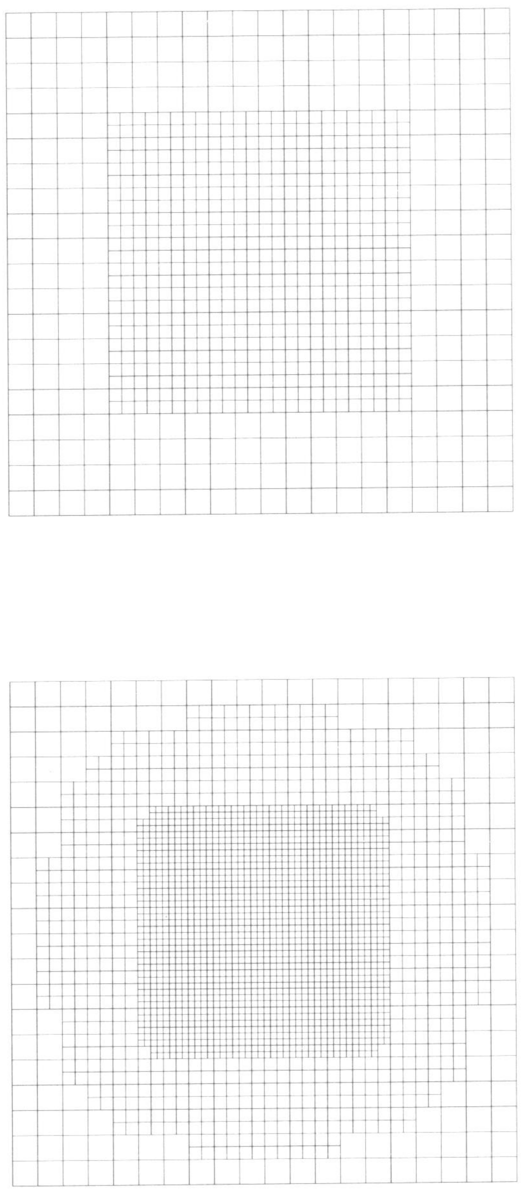


FIGURE 6.7 Problem II. The grids of the 2- and 3-level computation at $t = 1.0$.

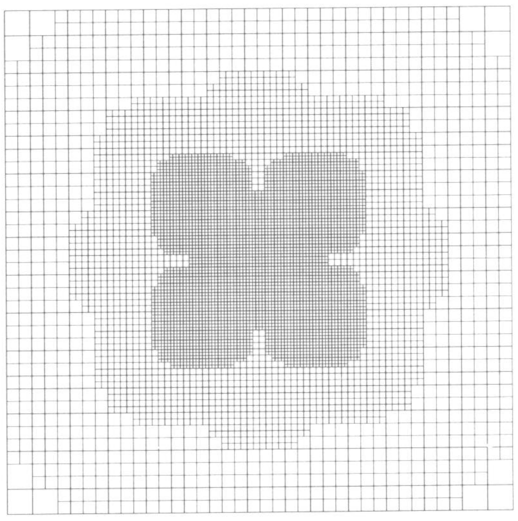


FIGURE 6.7 Continued. The grids of the 4-level computation at $t = 1.0$.

5.7. SUMMARY AND CONCLUDING REMARKS

In this paper we have discussed the application of a local uniform grid refinement method to systems of coupled PDEs. The main feature of local uniform grid refinement is that the PDEs are solved on a series of nested, uniform, Cartesian, increasingly finer subgrids covering only a part of the domain where the spatial error is high. The PDEs are solved on these subgrids in a consecutive manner, from coarse to fine. The location and size of the subgrids are automatically adjusted at discrete times in order to follow the movement of the steep fronts. The generation of subgrids is continued until sufficient spatial accuracy is reached.

An error analysis was performed for the local uniform grid refinement method applied to systems of coupled PDEs. It was shown that the global and local error components associated with each separate PDE can exhibit an entirely different behavior. With respect to the global error, this means that the global error components can carry over to future time points in a very different way from one PDE to another. The local space error components can show a different behavior for small time step sizes. A refinement strategy controlling the generation of finer subgrids was developed from the results of the error analysis. This strategy takes these

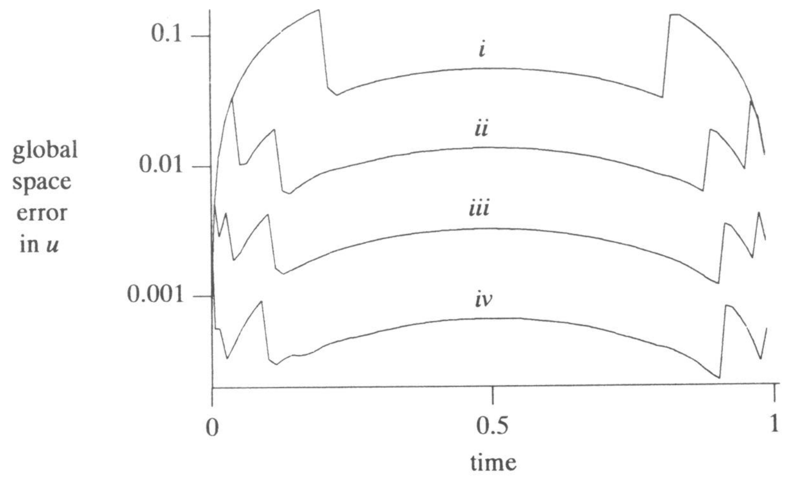
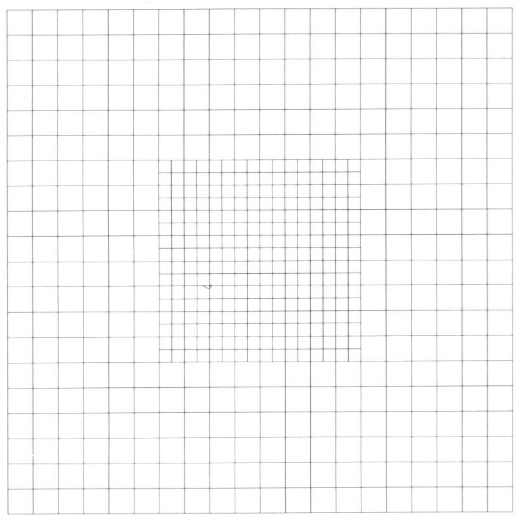


FIGURE 6.8 Problem III. The maximum global space error in u as a function of time. The maximum number of grid levels is 2 (i), 3 (ii), 4 (iii) and 5 (iv).

differences in behavior into account and is based on estimating and controlling the global space error. We have applied the method to three example problems, all involving a system containing a parabolic and an elliptic equation to test the refinement strategy. The observed convergence behavior of the global space error is comparable to uniform grid computations. We have also seen the predicted differences in behavior of the components of the global and local space error. Further, the global error estimates are fairly accurate and not many grid cells appear to be unnecessarily refined. Using both a fixed and a variable number of grid levels in time and a second-order space discretization, we have observed that when the tolerance value is decreased with a factor of four, the spatial accuracy also appears to improve with a factor of four.

We consider these results to be very satisfactory. However, we feel that testing on more difficult (nonlinear) problems needs to be done in order to fully appreciate this refinement strategy. Further, in the example problems where variable time step sizes were used, these step sizes were adapted in order to equidistribute a heuristic monitor. It would be desirable to implement a time step strategy based on (4.15) or (4.17). However, such a strategy only works well when the time error estimates are sufficiently accurate. In [7] we have already reported that when using the local uniform grid refinement method these time error estimates do not resemble the actual time error at all. Nevertheless, perhaps there is a remedy for this so that a time step



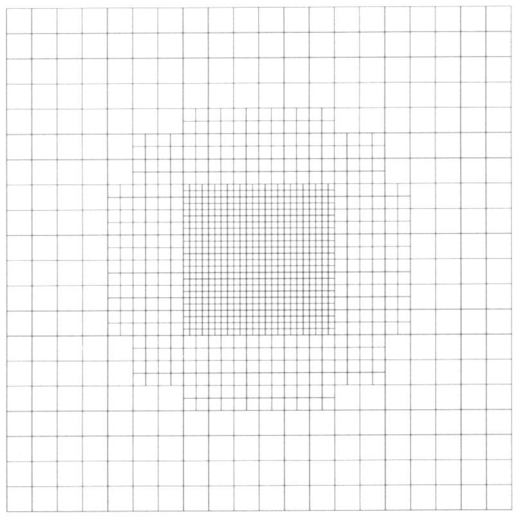
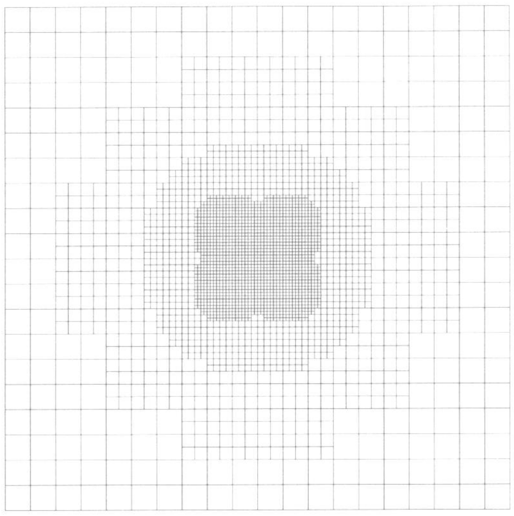


FIGURE 6.9 Problem III. The grids of the 2- and 3-level computation at $t = 0.5$.



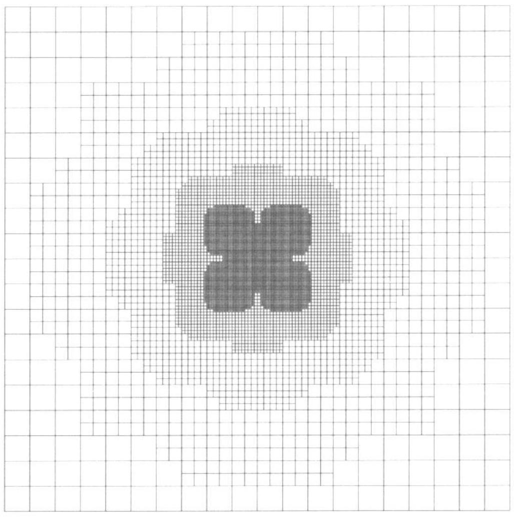


FIGURE 6.9 Continued. The grids of the 4- and 5-level computation at $t = 0.5$.

strategy can be developed which is just as successful as the refinement strategy in space.

REFERENCES

1. D.C. ARNEY and J.E. FLAHERTY (1989). An Adaptive Local Mesh Refinement Method for Time-Dependent Partial Differential Equations, *Appl. Numer. Math.*, 5, 257-274.
2. D.C. ARNEY and J.E. FLAHERTY (1990). An Adaptive Mesh-Moving and Local Refinement Method for Time-Dependent Partial Differential Equations, *ACM Trans. on Math. Softw.*, 16, 48-71.
3. M.J. BERGER and J. OLIGER (1984). Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, *J. Comput. Phys.*, 53, 484-512.
4. W.D. GROPP (1987). Local Uniform Mesh Refinement with Moving Grids, *SIAM J. Sci. Statist. Comput.*, 8, 292-304.
5. W.D. GROPP and D.E. KEYES (1992). Domain Decomposition with Local Mesh Refinement, *SIAM J. Sci. Comput.*, 13, 967-993.
6. J.E. FLAHERTY, P.K. MOORE and C. OZTURAN (1989). Adaptive Overlapping Grid Methods for Parabolic Systems, in *Adaptive Methods for Partial Differential Equations*, 176-193, ed. J.E. FLAHERTY, P.J. PASLOW, M.S. SHEPHARD, J.D. VASILAKIS, SIAM Publications, Philadelphia.
7. R.A. TROMPERT and J.G. VERWER (1991). A Static-Regridding Method for Two Dimensional Parabolic Partial Differential Equations, *Appl. Numer. Math.*, 8, 65-90.
8. R.A. TROMPERT and J.G. VERWER (1993). Analysis of the Implicit Euler Local Uniform Grid Refinement Method, *SIAM J. Sci. Comput.*, 18, 259-278.
9. R.A. TROMPERT and J.G. VERWER (1993). Runge-Kutta Methods and Local Uniform Grid Refinement, *Math. Comp.*, 60, 591-616.
10. J.G. VERWER and R.A. TROMPERT (1991). An Adaptive-Grid Finite-Difference Method for Time-Dependent Partial Differential Equations, in *Procs. 14th Biennial Conference on Numerical Analysis*, 267-284, ed. D.F. GRIFFITHS, G.A. WATSON, Pitman Research Notes in Mathematics Series 260, Dundee, Scotland.
11. J.G. VERWER and R.A. TROMPERT (1993). Analysis of Local Uniform Grid Refinement, *Appl. Numer. Math.*, 13, 251-270.

Chapter 6

Local Uniform Grid Refinement and Transport in Heterogeneous Porous Media

Ron Trompert
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

The application of an adaptive grid method to a mathematical model for unsteady flow, coupled with transport of solute in heterogeneous porous media is discussed. When the concentration of solute is large, the solute concentration profile in transport problems can show locally large gradients in space and in time. For this type of problems, an adaptive grid method can be greatly beneficial. We consider a method based on local uniform grid refinement, where integration takes place on a series of nested, local, uniform, finer-and-finer subgrids. These subgrids are created up to a level of refinement where sufficient spatial accuracy is reached and their location and shape is adjusted after each time step. The space domain is considered to be a rectangle and all grids in use are uniform and Cartesian. The interfaces, demarking the inhomogeneities, are assumed to coincide with cell edges in the numerical approximation. Special conditions are applied here, connecting the solutions on both sides of the interface. These interface conditions involve continuity of fluxes across the interfaces. The mesh refinement process and the variable time step sizes are controlled by heuristic error monitors.

to appear in: Adv. Water Resour.

6.1. Introduction

An adaptive grid finite-difference method is applied to a model for unsteady isothermal groundwater flow coupled with transport in heterogeneous porous media. The origin of this work lies in a safety assessment study on disposal of high-level radioactive wastes in rock salt formations, like salt domes. The numerical simulations of groundwater flow near these salt formations may provide insight in what might happen in the event of contaminants escaping from such a repository. The concentration of salt in the proximity of salt formations is known to be large and also in aquifers overlying these salt formations the salt content varies from fresh water to that of saturated brine [7]. It should be noted that the presence of a high concentration of salt in these natural situations gives rise to large concentration gradients as well. A typical situation one encounters is that of a sharp fresh-salt water interface that moves in time. A single, uniform space grid can be computationally very inefficient when solving the partial differential equations (PDEs) describing such problems because, to afford an accurate approximation, such a grid has to be very fine over the whole domain while a fine grid is only needed where a sharp front is located. Adaptive grid methods prove to be very useful here, since these methods refine the space grid only where it is really needed, hence, reducing the necessary CPU time.

The applied adaptive grid method is based on local uniform grid refinement. The main feature of local uniform grid refinement is that the PDEs are solved on a series of nested, uniform, finer-and-finer subgrids covering only a part of the domain. These subgrids are automatically adjusted at discrete times in order to follow the movement of large spatial variations. On each local subgrid a new initial boundary value problem is solved for one time step in a consecutive order, from coarse to fine. The generation of subgrids is continued until the spatial phenomena are described with sufficient accuracy.

Local uniform grid refinement methods have been proposed in a number of different varieties, applied to different kinds of PDEs. Here, we will not attempt to give a complete overview of the field. We will only sketch some varieties of the local uniform grid refinement method very briefly, and provide some references. The methods contained in [1-3, 5] are applied to hyperbolic PDEs and use explicit time stepping techniques. The method proposed by *Berger* and *Oliger* in [3] employs rectangular subgrids which may be skewed with respect to the co-ordinate axes in order to align with the steep region of the solution. Subgrids having the same cell sizes can partially overlap in this method. In [1], *Arney* and *Flaherty* developed a method very similar to the one in [3] except that the subgrids here are created by cellular refinement, meaning that the fine grid cells are properly nested within coarser grid cells. Hence, these subgrids have a piecewise polygonal shape.

Local uniform grid refinement is combined with grid movement in [2, 5]. In [5], a method proposed by *Gropp* uses subgrids which are rectangles having sides parallel to the co-ordinate axes and which are able to move as a whole with the moving steep fronts. In this method the subgrids are also allowed to overlap. In [2], *Arney* and *Flaherty* added grid movement to their method described in [1]. The grid points of the coarsest grid are able to move and the fine grid movement is induced by the movement of the coarsest grid. Local uniform grid refinement methods are also used to solve parabolic and elliptic PDEs in [6, 8-15] and involve the implicit solution of systems of equations. The subgrids in [8] are piecewise polygonal and the ones in [6] are rectangles. In both [8] and [6] domain decomposition is applied to improve the performance on parallel computers.

Our previous work on this type of adaptive grid method is contained in [9-15]. The subgrids in our method have a piecewise polygonal shape and do not overlap. Our method is a so-called 'static-regridding method' which means that no grid movement is applied during a time step. The refinement strategy controlling the generation of subgrids in [10, 13] is based on heuristic criteria while in [11, 12, 14, 15] it is underlied by a comprehensive error analysis which has resulted in a so-called refinement condition. This analysis was carried out for PDEs which after spatial discretization lead to a system of ordinary differential equations (ODEs). The aim of this refinement condition is that, once fulfilled, the overall spatial accuracy should be dominated by the spatial accuracy at the finest grid level. Due to the refinement condition, a convergence result could be proved in certain model situations which is similar to the result obtained for a single, uniform grid. In [9] the error analysis was carried out for systems of general coupled PDEs. These systems of PDEs do not necessarily lead to a system of ODEs after spatial discretization. A refinement strategy was derived based on the notion that the finest subgrid should contain the largest spatial errors. This strategy works quite satisfactorily but a rigorous convergence proof like the one mentioned previously can no longer be given.

In this work, the application of our version of the local uniform grid refinement method to transport problems in heterogeneous porous media is discussed. The adaptive grid method has been implemented in a code called MOORKOP. This code can handle systems of PDEs of the following type, defined on a rectangular domain

$$G(x,y,t,u,u_t,u_x,u_y,u_{xx},u_{xy},u_{yy}) = 0, \quad t > t_0. \tag{1.1a}$$

The boundary conditions may take the form

$$H(x,y,t,u,u_t,u_x,u_y) = 0, \quad t > t_0, \tag{1.1b}$$

and the initial conditions may be defined by

$$u(x,y,t_0) = u_0(x,y). \tag{1.1c}$$

The solution u may be vector valued. This general format was chosen to allow the user to solve not only transport problems in porous media but other PDEs as well. Moreover, in case transport problems are solved, the format (1.1) makes it easy for the user to implement various modifications to the basic equations of the model.

This work can be regarded as a sequel to the work reported in [13] in which this adaptive grid method was applied to transport problems in homogeneous porous media. Since in natural situations, soil properties such as permeability and transversal and longitudinal dispersivity can change abruptly from one region to another, MOORKOP has now been extended so that it can handle transport in porous media with such non-homogeneities. At these abrupt changes, interface conditions based on continuity of fluxes are applied to obtain consistent numerical approximations.

6.2. Outline of the adaptive grid method

The idea behind local uniform grid refinement is simple. Starting from a coarse base grid covering the whole domain, finer-and-finer uniform subgrids are created locally in a nested manner in regions of large spatial variations. Here, a set of interconnected grid cells, all having the same size, is called a *subgrid*. A set of subgrids having the same cell size is called a *grid level* or just *grid*. Hence, in our version of the local uniform grid refinement method, a grid level consists of a single subgrid or several disjunct, non-overlapping subgrids. A new initial boundary value problem is solved at each grid level separately in a consecutive order, from coarse to fine. The same time step size is used for all grids. The required initial values are defined by interpolation from the coarser grid level in which the refinement is embedded or taken from a grid level from the previous time step when available. Internal boundaries, i.e. subgrid boundaries lying in the interior of the domain, are treated as Dirichlet boundaries and values are also interpolated from the coarser grid level. Where the boundary of a fine subgrid coincides with the boundary of the domain, the prescribed boundary conditions are used. Except for the necessary initial and boundary conditions, all subgrids are independent of each other. Therefore, the subgrids are not patched into the coarser grids but are actually overlaying them. The generation of grid levels is continued until the spatial phenomena are described accurately enough by the finest grid. The fine grid cells are created by bisecting the sides of the cells of the coarser grid. This means that the refinement is cellular and that the subgrids have a piecewise polygonal shape.

During each time step the following operations are performed:

(1) *Solve PDEs on the coarse grid.*
(2) *If the desired accuracy in space or the maximum number of grid levels is reached then go to 8.*
(3) *Determine new finer grid level at forward time.*
(4) *Interpolate internal boundary values at forward time.*
(5) *Provide new initial values at backward time.*
(6) *Solve PDEs on new grid level, using the same time step.*
(7) *go to 2.*
(8) *Update the coarser grid solution using the finer grid values.*

Thus, for each time step the computation starts at the coarse base grid using the most accurate solution available, since coarser grid solution values are always updated by the finer grid solution.

For time integration we use implicit Euler for the first time step and the second-order two-step implicit BDF method with variable coefficients for the following time steps where variable step sizes are taken. Standard second-order central finite differences are used for space discretization and the interpolation, which is used for obtaining initial and boundary conditions, is linear. The discretization of the boundary conditions and the interface conditions are of first order. The unknowns in our difference scheme are located at vertices of cells. Hence, where the coarser grid is overlapped by a finer grid, the coarser grid nodes coincide with the finer grid nodes. The resulting systems of equations are solved by an adapted version of modified Newton's method in combination with the iterative linear solver BI-CGSTAB [16].

6.3. Strategies

6.3.1. Refinement strategy

The local uniform grid refinement method is a valuable method for solving PDEs with steep solutions because it can solve these PDEs just as accurately as on a very fine grid, but with considerably less computational effort, since the involved fine subgrids cover only a part of the domain. Moreover, it creates extra refinements when necessary and removes these when they are no longer needed. This refinement process is controlled by a refinement strategy. In [9, 11, 12, 14, 15], the refinement strategy is based on a comprehensive error analysis taking into account space discretization and interpolation error estimates. The aim of this strategy is to have the overall spatial accuracy dominated by the spatial accuracy at the finest grid level. When the number of grid levels is constant for all times, this strategy should lead to a spatial accuracy which is comparable to the one achieved with a single, uniform grid having cell sizes identical to those of the finest grid level in the adaptive grid method.

The success of such a refinement strategy is very much dependent on the accuracy of error estimates. It is clear that these error estimates can only be accurate when the solution is sufficiently smooth, i.e. it may be steep but it should be sufficiently differentiable in space. Since nonsmoothness in the boundary conditions, or even in the solution itself, is a well known phenomenon in flow and transport problems in porous media, the approach above was abandoned and replaced by a more heuristic approach. In [10, 13] the refinement strategy was based on a curvature monitor. This monitor is also used here because it is able to detect high-error regions. Further, this monitor can also detect kinks in the solution profile which occur at interfaces in porous media more quickly than a monitor based on the gradient of a solution. This implies that refinements at interfaces will be created much sooner. This is favorable for the accuracy of the computed solution, the approximated geometry of the interfaces, and also for solving the systems of nonlinear equations.

The error monitor can be regarded as a scaled approximation of $|\Delta x^2 \partial^2 u/\partial x^2| + |\Delta y^2 \partial^2 u/\partial y^2|$ for each solution component at every node. The value of this monitor at node j associated with solution component i is denoted as $ESTS_{i,j}$. At every grid node and for each solution component this monitor value is computed.

Let grid level 1 be the coarsest grid level and grid level 2 be the next finer grid level and so on. Suppose we have just completed a time step on grid level m. After this time step, the maximum values of $ESTS_{i,j}$ are computed over grid level m for each component i. These maxima are denoted as $ESTSmax_i$. If for some i,

$ESTSmax_i > TOLS$, then a new grid level $m+1$ is created within the current time step, provided that $m+1$ does not exceed the user-specified maximum number of grid levels. Here, TOLS is a user-defined tolerance. Grid level $m+1$ is the constructed as follows. For each i, for which $ESTSmax_i > TOLS$ holds, the cells around the nodes of grid level m where $ESTS_{i,j} > \frac{1}{4} \times TOLS$ will be subdivided in four identical cells. The set of these finer cells makes up grid level $m+1$, on which the current time step will now be repeated.

Finally, we have built in an extra condition to smooth the behavior of the code. Suppose that the maximum number of grid levels during the previous time step is *levtop* and that at grid level $m<levtop$, one has $ESTSmax_i \leq TOLS$. Although this means that a new finer grid level $m+1$ is actually not necessary, it will still be created when $ESTSmax_i > 0.9 \times TOLS$. This way fluctuation of the maximum number of grid levels from one time point to the next is likely to be avoided.

6.3.2. Time integration aspects

We have implemented the two-step BDF method of order two which we apply in the variable step size mode. The time derivative in (1.1) is then approximated as

$$U_t \approx \frac{U^n - a_1 U^{n-1} - a_2 U^{n-2}}{\theta_2 \Delta t_n}, \tag{3.1}$$

where

$$a_1 = \frac{(c+1)^2}{c^2+2c}, \quad a_2 = \frac{-1}{c^2+2c}, \quad \theta_2 = \frac{c+1}{c+2}, \tag{3.2}$$
$$c = \frac{\Delta t_{n-1}}{\Delta t_n}, \quad \Delta t_n = t_n - t_{n-1}.$$

Here, U_t represents the pointwise restriction of u_t to a space grid. Note that variable time stepping is a prerequisite for transport problems in porous media, as they can exhibit a highly distinct behavior in time. As starting formula we employ the one-step BDF method of order one (implicit Euler).

In the second-order BDF method, it would be appropriate to use the numerical estimates of the third time derivative of the solution for time step size control. This would work satisfactorily in standard applications where a grid without adaptation is used. However, one will encounter difficulties with such an estimator using a static-regridding method like the local uniform grid refinement method. This was already reported in [10]. These difficulties are due to the fact that the solution vector at the backward time points, present in the BDF formula, is obtained from interpolation from the coarser grid level, from computing the solution to (1.1) at the current grid level, and from assigning finer grid values to corresponding grid points at the current grid. Although these operations do not adversely affect the accuracy of the computed solution, they do introduce small 'discontinuities' in the solution which cause small stiff transient solution components in time. These small transients are quickly damped due to the stability of the BDF method. They are, however, seen by the local truncation error estimator which will in turn greatly overestimate the true local truncation error. This will lead to far too small time steps. In order to circumvent this problem we will use a time error monitor which is able to notice the transient behavior of the solution but does not see this background noise. This means that the values this monitor measures should be considerably larger than the background noise.

Here, the time step size is controlled by the time error monitor value which is a scaled approximation of $|\Delta t\, \partial u/\partial t|$ for each solution component at every node. This monitor worked quite satisfactorily in previous applications. The maximum value of this monitor, denoted as *ESTT*, is computed only over the interior grid nodes of each subgrid for reason of robustness of the code. We will not elaborate this further here. After a time step has been performed on all grid levels *ESTT* is computed over all grid levels. If this maximum exceeds a user-specified tolerance *TOLT*, then the time step is rejected, otherwise accepted. For each grid level a new time step size is predicted such that the predicted value of *ESTT* for the new time step is equal to $0.5 \times TOLT$. The minimum of these new time step size estimates is taken to be the time step size for the next time step. The rationale is that when a new time step is taken with this predicted value, *ESTT* will in general be approximately equal to $0.5 \times TOLT$ which implies that time step rejections are unlikely to occur. However, in case of a step rejection, the new time step size will be taken as 0.8 times this estimated value. In all cases, we require that the new time step size is not smaller than ⅓ times and not larger than 2 times the old time step size to avoid too large jumps in the step size selection. This strategy was adopted because of its performance in numerous previous applications. Finally, the new time step size is corrected with a small value to assure that the next output point is reached exactly.

6.3.3. Solution of the linear and nonlinear systems

Because we use an implicit integration method and treat PDEs like (1.1) fully coupled, we are facing the task of solving large coupled systems of nonlinear algebraic equations. Let the nonlinear system of equations to be solved be denoted as,

$$F(U) = 0. \tag{3.3}$$

In the standard modified Newton approach the linear system of equations

$$\begin{aligned} J(U^0)\delta^k &= -F(U^{k-1}), \\ U^k &= U^{k-1} + \delta^k, \end{aligned} \tag{3.4}$$

is subsequently solved, starting with $k=1$, until a stopping criterion is satisfied. Here

J is the Jacobian matrix, U^0 is the initial guess, and U^k is the k^{th} iterate. In our code, we use an adapted version of modified Newton in combination with preconditioned BI-CGSTAB [16] for iteratively solving the resulting system of linear equations. For any system of PDEs like (1.1), the required Jacobian matrix for the Newton process is computed in a completely automatic manner. In our code the elements of the Jacobian are estimated by a simple first-order difference formula, so that the user does not need to specify these. The reason why the standard modified Newton procedure has been adapted for the application of the local uniform grid refinement method is explained below.

When solving flow and transport problems in heterogeneous porous media one can encounter convergence problems. For example, when a subgrid is newly created or moves in space from one time point to the next, initial values for the new fine subgrid cells are interpolated from the next coarser subgrid solution which may be kinked due to an interface, leading to large interpolation errors. This way initial data is obtained which does not look like the fine grid solution to the PDEs at the backward time point, leading to a bad approximation of the Jacobian. To our experience, when the modified Newton iteration fails to converge because of this, the standard procedure of time step size reduction works very poorly, or not at all. For this reason the modified Newton procedure had to be adapted. The modifications we have made will now be elaborated upon.

The solution at the backward time point is taken as the initial guess for the finest grid level. With respect to the initial guess for the coarser grid levels, in spite of the fact that assigning of fine grid values to corresponding coarser grid nodes improves the accuracy of the solution at the coarser grid level (cf. Section 2, step 8), the updated coarser grid solution is usually not a very good initial guess for the solution at this grid at the future time point. Therefore, we also keep the original, not-updated solution at the backward time point in storage which is used as initial guess for the next time step. After this, the linear system (3.4) is generated and iteratively solved. In case this linear iteration process terminates unsuccessfully, (3.4) is generated all over again, employing a smaller time step size.

When (3.4) is solved at least twice (i.e. after two modified Newton iterations), we check for convergence and convergence speed. When the corresponding criterion is satisfied, the modified Newton stopping criterion is expected to be fulfilled within the user-specified maximum number of iterations. Note that the convergence(speed) criterion terminates a diverging as well as a slowly converging iteration process.

After this criterion has been fulfilled, we check if the modified Newton stopping criterion is satisfied. If this is the case then we are done, otherwise we proceed with the next iteration. In case that the convergence(speed) criterion is not satisfied, a new Jacobian is computed.

There are two ways to compute a new Jacobian. First, the Jacobian can be computed using the previous iterate U^{k-1} as initial guess and employing the same time step size. Second, we can compute the new Jacobian using the original initial guess U^0 with a reduced time step size, just as in the standard modified Newton approach. The way the Jacobian is calculated depends on a number of criteria. First, the number of new Jacobians using the same time step size during the whole iteration process is limited to a user-defined maximum. If this maximum is reached then the new Jacobian is computed with a reduced time step size. When a new Jacobian using the same time step size was already obtained during the previous iteration and the convergence(speed) criterion is still not satisfied, the new Jacobian is also calculated using a smaller time step size. Suppose that the last iteration where a new Jacobian was computed with the same time step size is denoted by j. We assume that when $\|F(U^{k-1})\|_\infty < \|F(U^{j-1})\|_\infty$, the iterate U^{k-1} is a 'better' solution to (3.3) than U^{j-1}. A new Jacobian is only computed using the same time step size if this is the case, and computed with a reduced time step size, otherwise.

This algorithm is more complicated than the standard modified Newton. Its behavior ranges from standard modified Newton to a genuine Newton-Raphson process. The idea behind it is that when the convergence criteria are not fulfilled, the iteration is not immediately repeated with a smaller time step size, like in the standard modified Newton approach, but a new Jacobian, based on the last accepted iterate and the same time step size, is tried first. Should this fail too, then the iteration is repeated with a smaller time step size.

The maximum number of Newton iterations and Jacobians should not be chosen too small, since this would lead to a premature termination of a converging iteration process. On the other hand, these numbers should not be chosen too large because a slowly converging iteration process can then carry on for a long time which is computationally inefficient. The maximum number of Newton iterations and Jacobians with the same time step size in our code are chosen to be 10 and 5 respectively which appeared to be reasonable choices in practice. When the time step size needs to be decreased, we take the new step size to be ¼ times the previous one.

6.4. Model of brine transport in porous media

In this section the mathematical model of brine transport used to solve the example problem (cf. Section 6) is described. Following *Trompert*, *Verwer* and *Blom* [13], we consider a model for unsteady, isothermal, single-phase, two-component, saturated flow in a porous medium in two space dimensions. This model contains two conservation laws, namely one for the mass of the whole fluid, i.e. water and salt, and one for the mass of salt only. The mass conservation of the fluid supplemented with Darcy's law for the velocity field is given by

$$\frac{\partial}{\partial t}(n\rho) + \nabla.(\rho\mathbf{q}) = 0, \quad \mathbf{q} = -\frac{k}{\mu}(\nabla p - \rho\mathbf{g}), \tag{4.1}$$

where n is the porosity of the porous medium, ρ is the mass density and $\mathbf{q}$ the velocity vector of the fluid. Note that here the velocity of the solid phase is neglected. The permeability of the porous medium is denoted by k, μ is the dynamic viscosity, p pressure and $\mathbf{g}$ the gravity vector. The mass conservation law of salt and Fick's law for the dispersive mass flux are given by

$$\frac{\partial}{\partial t}(n\rho\omega) + \nabla.(\rho\omega\mathbf{q} + \mathbf{J}) = 0, \quad \mathbf{J} = -\rho n\mathbf{D}\nabla\omega, \tag{4.2}$$

respectively, where ω is the concentration of salt and $\mathbf{J}$ the dispersive mass flux vector. $\mathbf{D}$ is the 2×2 dispersion tensor defined by

$$n\mathbf{D} = (nd_m + \alpha_T|\mathbf{q}|)\mathbf{I} + (\alpha_L - \alpha_T)\frac{\mathbf{q}\mathbf{q}^T}{|\mathbf{q}|}, \quad |\mathbf{q}| = (\mathbf{q}^T\mathbf{q})^{1/2}, \tag{4.3}$$

where α_L denotes the longitudinal and α_T the transversal dispersivity and d_m the molecular diffusion coefficient. $\mathbf{I}$ is the 2×2 identity matrix. The soil properties in this model are n, d_m, α_L, α_T and k. Temperature and compressibility effects are neglected in this model, as well as sources, sinks and deformation of the porous medium. To complete the model we need an equation of state for the fluid mass density ρ and an expression for the dynamic viscosity μ which depends on the concentration of salt:

$$\rho = \rho_0 \exp(\gamma\omega), \tag{4.4}$$

$$\mu = \mu_0(1 + 1.85\omega - 4.10\omega^2 + 44.50\omega^3), \tag{4.5}$$

where ρ_0 and μ_0 are the reference density and dynamic viscosity and γ is a coefficient obtained from laboratory experiments.

In cases of a low salt concentration (4.1) and (4.2) are only weakly coupled and can be solved independently. The flow can then be regarded as independent from the density gradients caused by differences in the salt concentration since these gradients prove to be negligible. However, we consider cases of high salt concentration, in which case the flow is no longer independent from the density gradients, so these equations should be solved together. With this model we have followed *Hassanizadeh* and *Leijnse* [7] in the description of brine transport, except for Darcy's law and Fick's law. In this paper these laws are used in their classical formulation, valid for low concentration cases.

Using p and ω as independent variables, we have recasted equations (4.1), (4.2) in the form

$$\begin{aligned} &-\gamma\nabla.\mathbf{J} + \rho\nabla.\mathbf{q} = 0, \\ &\rho n\frac{\partial\omega}{\partial t} + \rho\mathbf{q}.\nabla\omega + \nabla.\mathbf{J} = 0, \end{aligned} \tag{4.6}$$

which is obtained after some elementary calculations. At this stage we note that this model fits into format (1.1) and can, within the limits of (1.1), be modified by the user of the code. For example, one can add a temperature equation or use different formulations for Darcy's law and/or Fick's law, or by add compressibility effects, etcetera.

6.5. Interface conditions

In this section we explain the interface conditions. These conditions are based on continuity of fluxes across cell edges. Although we will only discuss the interface conditions using the mathematical model of brine transport from the previous section, it is straight forward to derive interface conditions for other transport problems in heterogeneous porous media.

The soil properties in heterogeneous porous media can show abrupt changes from one region to another. Moreover, across these interfaces, i.e. where the sudden changes occur, p and ω are continuous but their profiles may be kinked. We will assume that, mathematically, these soil properties are piecewise constant functions and that p and ω are both continuous functions, not differentiable in space, at an interface. This means that in order to get consistent numerical approximations, we have to take care that numerical differentiation does not take place across such an interface. Therefore, the numerical solution at an interface is obtained by fulfilling interface conditions which connect the solution on both sides of the interface and involve only one-sided difference schemes. Since (4.1) and (4.2) represent two conservation laws, it is natural to impose continuity of the spatial fluxes $\rho\mathbf{q}.\mathbf{n}$ and $(\rho\omega\mathbf{q} + \mathbf{J}).\mathbf{n}$ at interfaces as interface conditions, where $\mathbf{n}$ is a unit vector locally perpendicular to the interface. It suffices to impose continuity of $\mathbf{q}.\mathbf{n}$ and $\mathbf{J}.\mathbf{n}$, since ρ and ω are both continuous functions.

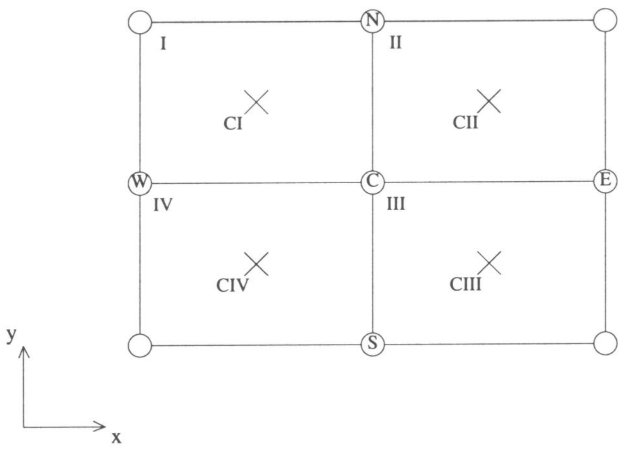


FIGURE 5.1. Four arbitrary grid cells with cell edges, parallel to the co-ordinate axes.

Consider the four grid cells shown in Figure. 5.1, numbered I through IV, with cell edges parallel to the co-ordinate axes. First, the soil properties are evaluated in all cell centers and are supposed to be constant over each cell. Hence, the interfaces are assumed to coincide with cell edges in the numerical approximation. The means that the interfaces are approximated by piecewise vertical and horizontal lines.

When the soil properties are constant over these four cells then none of these cells are intersected by an interface and (4.6) is discretized at grid node C using the standard second-order central finite differences in space. Now suppose that, for example, the soil properties in CI are different from those in CII. Then the component of **q** and **J** in x-direction which is perpendicular to the cell edge, separating the upper left cell I from the upper right cell II, must be continuous. This cell edge is denoted as CN. From (4.1)-(4.3) we have,

$$\begin{aligned}
q_1 &= -\frac{k}{\mu}(p_x - \rho g_1),\\
q_2 &= -\frac{k}{\mu}(p_y - \rho g_2),\\
J_1 &= -\rho n D_{11}\omega_x - \rho n D_{12}\omega_y,\\
nD_{11} &= nd_m + \alpha_T|\mathbf{q}| + (\alpha_L - \alpha_T)\frac{q_1^2}{|\mathbf{q}|},\\
nD_{12} &= (\alpha_L - \alpha_T)\frac{q_1 q_2}{|\mathbf{q}|},
\end{aligned} \tag{5.1}$$

where q_1, J_1 and g_1 are the components in x-direction of **q**, **J** and **g**, respectively, and nD_{11} and nD_{12} are elements of the first row of the dispersion tensor $n\mathbf{D}$; q_2 and g_2 are the components in y-direction of **q** and **g**. The derivatives of (5.1) are discretized using the grid nodes N, S, E, W and C, which yields a first-order accurate discretization. The fluxes q_1 and J_1 on the left and right hand side of CN are denoted as $q_{1,CN,I}$, $J_{1,CN,I}$ and $q_{1,CN,II}$, $J_{1,CN,II}$, respectively. They are now approximated as

$$\begin{aligned}
q_{1,CN,I} &= -\frac{k_{CI}}{\mu_C}(\frac{p_C - p_W}{\Delta x} - \rho_C g_1),\\
J_{1,CN,I} &= -\rho_C n D_{11,CN,I}\frac{\omega_C - \omega_W}{\Delta x} - \rho_C n D_{12,CN,I}\frac{\omega_N - \omega_C}{\Delta y},\\
q_{1,CN,II} &= -\frac{k_{CII}}{\mu_C}(\frac{p_E - p_C}{\Delta x} - \rho_C g_1),\\
J_{1,CN,II} &= -\rho_C n D_{11,CN,II}\frac{\omega_E - \omega_C}{\Delta x} - \rho_C n D_{12,CN,II}\frac{\omega_N - \omega_C}{\Delta y},
\end{aligned} \tag{5.2}$$

where

$$\begin{aligned}
nD_{11,CN,I} &= nd_{m,CI} + \alpha_{T,CI}|\mathbf{q}_{CN,I}| + (\alpha_{L,CI}-\alpha_{T,CI})\frac{q^2_{1,CN,I}}{|\mathbf{q}_{CN,I}|},\\
nD_{12,CN,I} &= (\alpha_{L,CI}-\alpha_{T,CI})\frac{q_{1,CN,I}\,q_{2,CN,I}}{|\mathbf{q}_{CN,I}|},\\
nD_{11,CN,II} &= nd_{m,CII} + \alpha_{T,CII}|\mathbf{q}_{CN,II}| + (\alpha_{L,CII}-\alpha_{T,CII})\frac{q^2_{1,CN,II}}{|\mathbf{q}_{CN,II}|},\\
nD_{12,CN,II} &= (\alpha_{L,CII}-\alpha_{T,CII})\frac{q_{1,CN,II}\,q_{2,CN,II}}{|\mathbf{q}_{CN,II}|}, \qquad (5{,}3)\\
q_{2,CN,I} &= -\frac{k_{CI}}{\mu_C}\left(\frac{p_N-p_C}{\Delta y} - \rho_C g_2\right),\\
q_{2,CN,II} &= -\frac{k_{CII}}{\mu_C}\left(\frac{p_N-p_C}{\Delta y} - \rho_C g_2\right),\\
|\mathbf{q}_{CN,I}| &= (q^2_{1,CN,I}+q^2_{2,CN,I})^{1/2}, \qquad |\mathbf{q}_{CN,II}| = (q^2_{1,CN,II}+q^2_{2,CN,II})^{1/2}.
\end{aligned}$$

Here $q_{2,CN,I}$ and $q_{2,CN,II}$ represent velocities parallel to CN and $nD_{11,CN,I}$, $nD_{12,CN,I}$, $nD_{11,CN,II}$, $nD_{12,CN,II}$ are the elements of the first row of the dispersion tensor on the two sides of CN. Constants like k_{CII} and $\alpha_{L,CII}$ denote the permeability and longitudinal dispersivity at cell II and entries like, for example, ρ_C and μ_C are the mass density and the dynamic viscosity in C. Continuity of $\mathbf{q}.\,\mathbf{n}$ and $\mathbf{J}.\,\mathbf{n}$ across CN yields the following system of flux continuity equations for p and ω in C

$$\begin{aligned}
q_{1,CN,I} - q_{1,CN,II} &= 0, \qquad (5.4)\\
J_{1,CN,I} - J_{1,CN,II} &= 0.
\end{aligned}$$

When not only CN is an interface but also CW, CE or CS then the flux continuity equations are generated for each interface. The equations we then solve is the sum of these flux continuity equations.

6.6. Numerical illustration

An example problem is presented dealing with the displacement of fresh water by brine in a vertical column, filled with a porous medium and measuring one by one meter. Here we assume that $\mathbf{g} = (0,-g)^T$. The values of the parameters are chosen as

$$\begin{aligned}
n &= 0.4, \quad d_m = 0\,m^2.s^{-1}, \quad \rho_0 = 10^3\,kg.m^{-3}, \quad p_0 = 10^5\,N.m^{-2}, \qquad (6.1)\\
\gamma &= \log(2.0), \quad g = 9.81\,m.s^{-2}, \quad \mu_0 = 10^{-3}kg.m^{-1}.s^{-1}..
\end{aligned}$$

In the vertical column considered, there are four different regions, indicated as I through IV. This is shown in Figure 6.1. Each of these regions has its own permeability and longitudinal and transversal dispersivity. These are given below in (6.3).

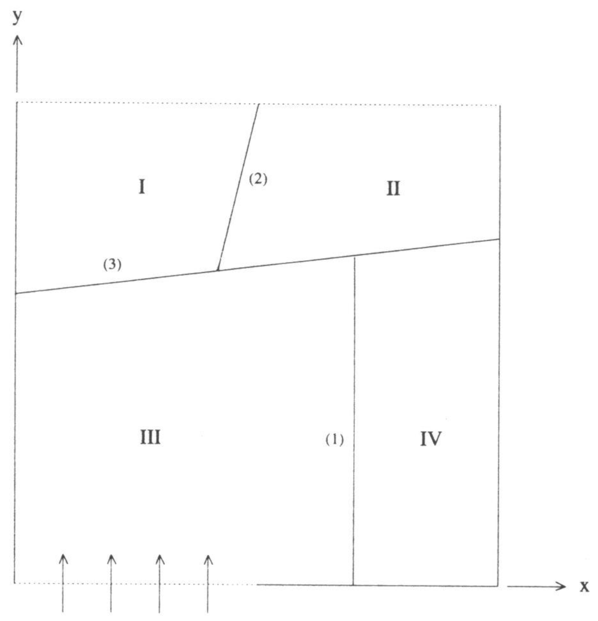


FIGURE 6.1. The vertical column.

The column is completely open at the top and only half open at the bottom. The vertical sides are closed. The initial values, boundary conditions are:

$$
\begin{aligned}
&p(x,y,0) = p_0 + (1-y)\rho_0 g, \quad \omega(x,y,0) = 0, \quad 0\,m < x,y < 1\,m,\\
&q_1 = 0\,m.s^{-1}, \quad \omega_x = 0\,m^{-1}, \quad x = 0,1\,m \text{ and } 0\,m < y < 1\,m,\\
&q_2 = 10^{-4}\,m.s^{-1}, \quad \omega = 0.25 \times (1-\exp(-10t)), \qquad (6.2)\\
&\qquad 0\,m < x \le 0.5\,m \text{ and } y = 0\,m,\\
&q_2 = 0\,m.s^{-1}, \quad \omega = 0, \quad 0.5\,m < x < 1\,m \text{ and } y = 0\,m,\\
&p = p_0, \quad \omega_y = 0\,m^{-1}, \quad 0\,m < x < 1\,m \text{ and } y = 1\,m.
\end{aligned}
$$

The soil properties are given by:

$$
\begin{aligned}
&\text{Region I:} \quad k = 10^{-13}\,m^2, \quad \alpha_L = 0.008\,m, \quad \alpha_T = 0.0016\,m,\\
&\text{Region II:} \quad k = 10^{-15}\,m^2, \quad \alpha_L = 0.005\,m, \quad \alpha_T = 0.0010\,m, \qquad (6.3)\\
&\text{Region III:} \quad k = 10^{-10}\,m^2, \quad \alpha_L = 0.010\,m, \quad \alpha_T = 0.0020\,m,
\end{aligned}
$$

$$\text{Region IV:} \quad k = 10^{-13}\, m^2, \quad \alpha_L = 0.008\, m, \quad \alpha_T = 0.0016\, m.$$

The interfaces are defined as:

$$\begin{aligned} &1: \quad x{=}0.7\, m, \\ &2: \quad x{=}0.3\, m + 0.2 \times y, \\ &3: \quad y{=}0.6\, m + 0.1 \times x. \end{aligned} \tag{6.4}$$

Saturated brine is injected into the column at the opening in the bottom and a steep front in the salt concentration will develop, moving slowly towards the top of the column. At first, the front will move completely past the interface on its right hand side. So, initially there will be almost no penetration of salt into region IV and a very sharp transition in the salt concentration arises at the interface between III and IV. The front smoothes while moving. Later on, the front will pass the interface between III and I and will move into region I. Much later the salt penetrates II and IV from III at approximately the same time. First the salt penetrates IV at its top left corner and later at the entire interface (1). Steady state is reached when eventually the saturated brine has spread out over the entire domain.

We have computed the solution to this problem with two and three grid levels, of which the coarsest is a 20×20 grid. We have chosen $TOLT = 0.1$ and $TOLS = 0.25$ for both cases. The salt concentration is shown in the Figures 6.2 through 6.5. The lower boundary of these figures corresponds with the top of the column. We have also computed the solution on a single, uniform 40×40 and 80×80 grid for comparison. The absolute value of the difference between the salt concentration obtained with the uniform grid computation and the adaptive grid computation is plotted in the Figures 6.6 and 6.7.

It appears that the maxima of these differences of 0.05 in both cases are rather large. A reason for this could be that the refinement strategy is not restrictive enough, i.e. too few grid cells are refined. Moreover, the spatial error monitor bears no relationship with the true numerical errors which means that it is possible that cells that should have been refined are not refined. Both of these factors contribute to a larger numerical error. Further, the difference scheme we have used, the way internal grid boundaries are treated and the interpolation and updating procedure can be a cause, since, neither one is conservative. Applying a control volume scheme and observing the conservation property at grid interfaces might help here. Nevertheless, the real reason for the large differences is still unclear to us.

We have shown 3D plots of the salt concentration at the top and the pressure at the bottom of Figure 6.8 at $t{=}4000\, s$, computed on the 40×40 grid. The kinked solution profile at some interfaces are clearly visible here. Note that at the interface (1) there is a sharp kink in the salt concentration and no visible kink in the pressure. We have also compared the necessary CPU times of the adaptive grid and uniform grid computations. These are compared in Table 6.1. We can see that the two-grid and three-grid computations are 1.8 and 4.4 times faster than the corresponding

# of grid levels	single grid	CPU time *sec.*
2		3332
	40×40	5995
3		9097
	80×80	40441

TABLE 6.1. CPU times of adaptive grid and uniform grid computations.

grid level	# of Newton iterations	# of Jacobians
1	973	294
2	1051	291

TABLE 6.2. # of Newton iterations and Jacobians, needed for the two-grid computation. The number of time steps is 290.

grid level	# of Newton iterations	# of Jacobians
1	1074	322
2	1175	326
3	1189	318

TABLE 6.3. # of Newton iterations and Jacobians, needed for the three-grid computation. The number of time steps is 318.

uniform grid computations. These measurements were performed on a Silicon Graphics INDIGO workstation. The Tables 6.2 and 6.3 contain information about Newton iteration process of both adaptive grid computations. We can see that for both the two- and the three-grid level computations the number of Jacobians is only moderately larger than the number of time steps. Hence, our Newton iteration strategy performs well in combination with the local uniform grid refinement method for this case.

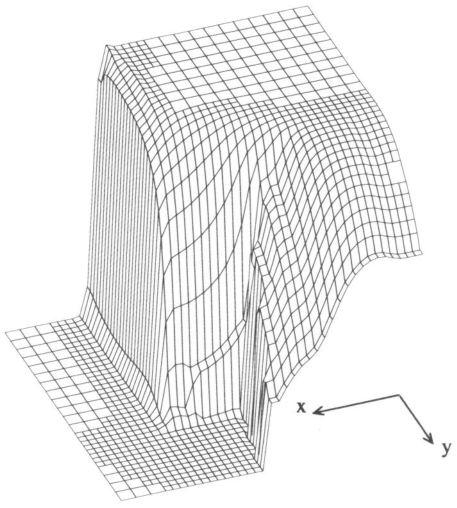


FIGURE 6.2. The salt concentration with two grid levels at t=4.10^3.

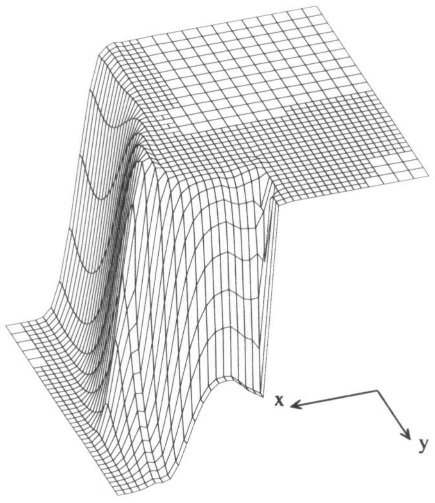


FIGURE 6.3. The salt concentration with two grid levels at t=6.10^4.

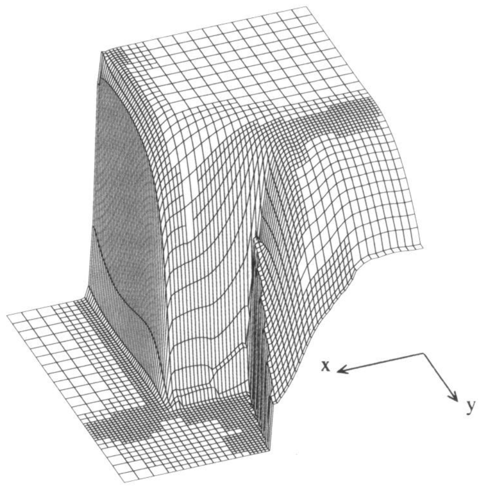


FIGURE 6.4. The salt concentration with three grid levels at t=4.10^3.

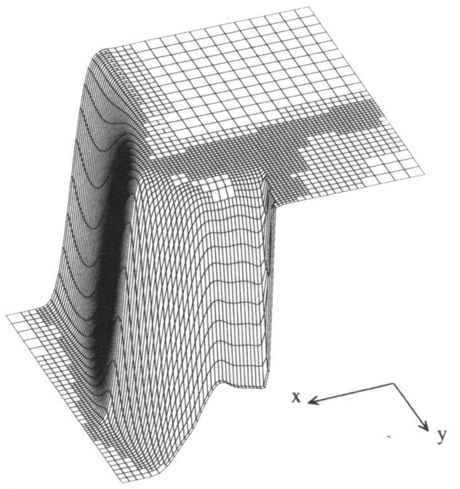


FIGURE 6.5. The salt concentration with three grid levels at t=6.10^4.

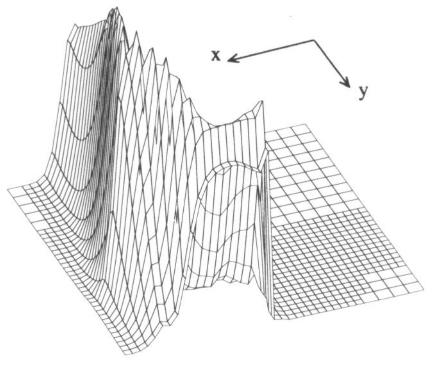


FIGURE 6.6. The absolute differences in salt concentration. The 40×40 grid solution compared to the two-level adaptive grid solution at t=6.10^4. The maximum is 0.05.

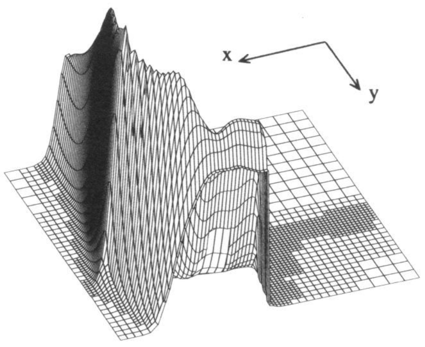
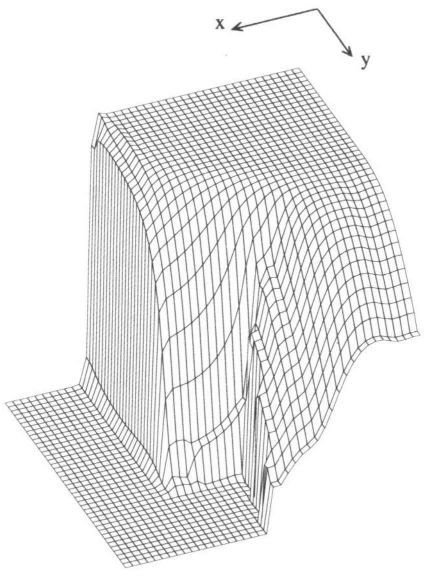


FIGURE 6.7. The absolute differences in salt concentration. The 80×80 grid solution compared to the three-level adaptive grid solution at t=6.10^4. The maximum is 0.05.



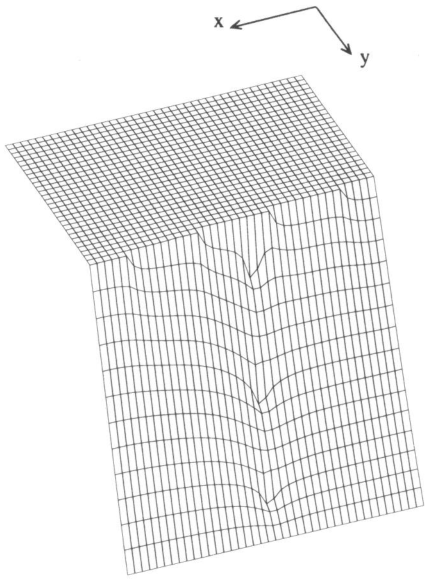


FIGURE 6.8. The salt concentration and the pressure at $t = 4.10^3$ with a 40×40 grid.

6.7. Summary and concluding remarks

In this paper we have discussed the application of a code using local uniform grid refinement, to transport problems in heterogeneous porous media. For such problems, where locally steep fronts in the solute concentration occur, adaptive grid methods are valuable. They can compute a solution to these problems with locally the same resolution as on a very fine uniform grid, but with less computational costs.

With respect to the modern computer architectures we note that it is easier to obtain large speed ups on a vector/parallel computer with a rectangular uniform grid than with an adaptive grid method. However, according to [4], we can obtain considerable gains with an adaptive grid method too. On top of that, there will always be cases in which it is very difficult or even impossible to perform accurate uniform grid computations due to memory requirements, especially in three space dimensions.

In natural circumstances, the soil properties of the porous medium can change very suddenly from one region to another. At these sudden changes the profile of the pressure or the solute concentration may be kinked. Consequently, interface conditions, implying continuity of fluxes across these interfaces and involving only one-sided difference schemes, have been applied here to obtain consistent numerical approximations. The 'numerical' interfaces are supposed to coincide with grid cell edges. Further, compared to our previous publication [13], the modified Newton method for solving the systems of nonlinear equations has been adapted to increase the robustness of the code.

The results of the test problem indicates that the solution computed with the local uniform grid refinement method requires less costs than a comparable uniform grid method. However, we have observed considerable local differences between the uniform grid and the adaptive grid solution. The cause of this is still unclear to us and should be investigated. The results also indicate that the adapted modified Newton method and the linear iterative solver BI-CGSTAB work satisfactorily for the example problem. Nevertheless we think that a warning is appropriate here. Although the adaptation of the modified Newton method has improved the robustness of the code considerably, there still is a possibility that the code breaks down, simply because the (partially) interpolated initial guess for the iteration process (cf. Section 3) is too far away from the solution of the system of nonlinear equations at hand. A remedy to this could be to create at all times finer grids at interfaces, whether or not required by the space error monitor.

Acknowledgement

This work was carried out as a part of contract research by order of the Laboratory for Soil and Groundwater Research of RIVM - the Dutch National Institute of Public Health and Environmental Protection - in connection with project 'Locatiespecifieke Modelvalidatie' 725205. Financial support for this project was provided by the Dutch Ministry of Economic Affairs.

References

1. D.C. Arney and J.E. Flaherty (1989). An Adaptive Local Mesh Refinement Method for Time-Dependent Partial Differential Equations, *Appl. Numer. Math.*, 5, 257-274.
2. D.C. Arney and J.E. Flaherty (1990). An Adaptive Mesh-Moving and Local Refinement Method for Time-Dependent Partial Differential Equations, *ACM Trans. on Math. Softw.*, 16, 48-71.
3. M.J. Berger and J. Oliger (1984). Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, *J. Comput. Phys.*, 53, 484-512.
4. J.G. Blom and J.G. Verwer (1993). *VLUGR2: A Vectorized Local Uniform Grid Refiniment Method for PDEs in 2D,* Report NM-R9306, Centre for Mathematics and Computer Science, Amsterdam.
5. W.D. Gropp (1987). Local Uniform Mesh Refinement with Moving Grids, *SIAM J. Sci. Statist. Comput.*, 8, 292-304.
6. W.D. Gropp and D.E. Keyes (1992). Domain Decomposition with Local Mesh Refinement, *SIAM J. Sci. Comput.*, 13, 967-993.
7. S.M. Hassanizadeh and T. Leijnse (1988). On the Modeling of Brine Transport in Porous Media, *Water Resources Research*, 24, 321-330.
8. J.E. Flaherty, P.K. Moore and C. Ozturan (1989). Adaptive Overlapping Grid Methods for Parabolic Systems, in *Adaptive Methods for Partial Differential Equations*, 176-193, ed. J.E. Flaherty, P.J. Paslow, M.S. Shephard, J.D. Vasilakis, SIAM Publications, Philadelphia.
9. R.A. Trompert (1993). Local Uniform Grid Refinement and Systems of Coupled Partial Differential Equations, *Appl. Numer. Math.*, 12, 331-355.
10. R.A. Trompert and J.G. Verwer (1991). A Static-Regridding Method for Two Dimensional Parabolic Partial Differential Equations, *Appl. Numer. Math.*, 8, 65-90.
11. R.A. Trompert and J.G. Verwer (1993). Analysis of the Implicit Euler Local Uniform Grid Refinement Method, *SIAM J. Sci. Comput.*, 18, 259-278.
12. R.A. Trompert and J.G. Verwer (1993). Runge-Kutta Methods and Local Uniform Grid Refinement, *Math. Comp.*, 60, 591-616.
13. R.A. Trompert, J.G. Verwer, and J.G. Blom (1993). Computing Brine Transport in Porous Media with an Adaptive-Grid Method, *Int. J. Numer. Meths. in Fluids*, 16, 43-63.
14. J.G. Verwer and R.A. Trompert (1991). An Adaptive-Grid Finite-Difference Method for Time-Dependent Partial Differential Equations, in *Procs. 14th Biennial Conference on Numerical Analysis*, 267-284, ed. D.F. Griffiths, G.A. Watson, Pitman Research Notes in Mathematics Series 260, Dundee, Scotland.
15. J.G. Verwer and R.A. Trompert (1993). Analysis of Local Uniform Grid Refinement, *Appl. Numer. Math.*, 13, 251-270.
16. H.A. van der Vorst (1992). BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsymmetric Linear Systems, *SIAM J. Sci. Statist. Comput.*, 13(2), 631-644.

Samenvatting

Veel processen in de natuur worden beschreven door partiële differentiaalvergelijkingen (PDV's). Deze processen kunnen gesimuleerd worden door de bijbehorende PDV's op te lossen. The complexiteit van deze PDV's vereist echter dat deze vergelijkingen numeriek moeten worden opgelost. Om voor een PDV een numerieke oplossing te krijgen wordt de PDV benaderd (gediscretiseerd) op een verzameling discrete punten (rooster) wat resulteert in een stelsel algebraische vergelijkingen. Door dit stelsel op te lossen wordt een numerieke oplossing berekend die een benadering is van de oplossing van de PDV. Als een tijdsafhankelijk probleem wordt opgelost dan wordt, beginnende bij de bekende oplossing op het begin tijdstip, het stelsel algebraische vergelijkingen opgelost om een numerieke oplossing te berekenen op een iets later tijdstip. Dit proces, wat ook wel tijdstappen wordt genoemd, wordt herhaald totdat het eind tijdstip is bereikt.

Veel PDV's hebben oplossingen die snel variëren in ruimte en tijd. Ruw weg kan men zeggen dat de variatie in de oplossing van roosterpunt tot roosterpunt de nauwkeurigheid bepaalt waarmee de numerieke oplossing de oplossing van de PDV benaderd. Hoe groter de variatie in de oplossing in ruimte en tijd is, des te fijner moet het rooster zijn om een nauwkeurige oplossing te krijgen. In veel gevallen is de variatie in de oplossing niet over het hele domein groot maar alleen lokaal. Om in dit soort gevallen de bijbehorende PDV's nauwkeurig en op een efficiënte manier op te lossen zou het rooster fijn moeten zijn waar de variatie groot is en grover waar de variatie minder groot is. Adaptieve roostermethoden zijn methoden die dit proberen te verwezenlijken.

Het onderwerp van dit proefschrift is een adaptieve roostermethode genaamd de *lokale uniforme roosterverfijningsmethode*. Het voornaamste kenmerk van lokale uniforme roosterverfijning is dat de PDV's opgelost worden op een aantal in toenemende mate fijnere lokale sub-roosters die gegenereerd worden in gebieden waar de variatie in de oplossing groot is. Op elk sub-rooster afzonderlijk wordt in de volgorde van grof naar fijn de PDV's opgelost voor één tijdstap. De lokatie en vorm van deze sub-roosters wordt op discrete tijdstippen aangepast om de beweging van de gebieden met grote variaties te kunnen volgen. De generatie van de sub-roosters wordt gestuurd door een verfijningsstrategie die gebaseerd kan zijn op een schatting van de fout die gemaakt wordt of op de waarde van een foutmonitor. Zo'n foutmonitor kan bijvoorbeeld afhankelijk zij van de helling of de kromming van de oplossing. Een verfijningsstrategie gebaseerd op een foutmonitor vergt minder rekentijd dan een strategie gebaseerd op foutschattingen. Niettemin geeft een verfijningsstrategie gebaseerd op foutschattingen vaak nauwkeuriger resultaten dan de foutmonitor strategie. Dit komt omdat er geen relatie bestaat tussen een foutmonitor en de fout die gemaakt wordt. Om deze reden zal een foutschatting strategie vaak betere sub-roosters creëren dan een foutmonitor strategie. De lokale uniforme roosterverfijningsmethode zoals beschreven in dit proefschrift richt zich op het in de ruimte verfijnen van het rooster waardoor de ruimtelijke component van de fout kan worden beheerst.

Dit proefschrift bevat een vijftal tijdschriftartikelen. Op hoofdstuk 1 na bevat elk hoofdstuk een artikel. De hoofdstukken 2 en 6 zijn toegepast van aard terwijl de hoofdstukken 3,4 en 5 wat fundamenteler zijn. Deze laatstgenoemde hoofdstukken vormen de basis van dit proefschrift. In deze hoofdstukken worden verfijningsstrategieën gebaseerd op foutschattingen ontwikkeld uit foutanalyses. Deze analyses hebben betrekking op verschillende tijdstapschema's en verschillende typen PDV's. Een verfijningsstrategie gebaseerd op een foutmonitor is gebruikt in de hoofdstukken 2 en 6. In hoofdstuk 6 wordt de toepassing besproken van de lokale uniforme roosterverfijningsmethode op de simulatie van transport door heterogene poreuze media. Het werk dat aan dit artikel ten grondslag ligt is uitgevoerd in opdracht van Rijksinstituut voor Volksgezondheid en Milieuhygiène (RIVM). In het kader van dit projekt is de lokale uniforme roosterverfijningsmethode geimplementeerd in een code genaamd MOORKOP. Deze code is ontwikkeld voor een vrij brede klasse van partiële differentiaalvergelijkingen waaronder ook transport problemen in heterogene poreuze media vallen.

Stellingen behorende bij het proefschrift

Local Uniform Grid Refinement for Time-Dependent Partial Differential Equations

van R.A. Trompert

1. Veronderstel dat bij het oplossen van een tijdsafhankelijke partiële differentiaalvergelijking de impliciete Euler-methode gebruikt wordt voor het tijdstappen. Als dan het numerieke schema bij gebruik van één uniform rooster stabiel is dan zal dit bij de lokale uniforme roosterverfijningsmethode ook het geval zijn als de gebruikte interpolatie van begin- en randwaarden lineair is.
Hoofdstuk 3 van dit proefschrift.

2. Veronderstel dat het aantal gebruikte verfijningsniveaus bij de lokale uniforme roosterverfijningsmethode constant is in de tijd en dat de impliciete Euler methode wordt gebruikt voor het tijdstappen. Als vervolgens aan de aannames A4 en A5 uit hoofdstuk 3 wordt voldaan, de interpolatie lineair is en bij het creëren van de lokale sub-roosters de ongelijkheid (6.9) uit hetzelfde hoofdstuk in acht wordt genomen, dan is de begrenzing van de lokale plaatsfout bij gebruik van de lokale uniforme roosterverfijningsmethode en bij gebruik van één uniform grid vergelijkbaar.

3. Veronderstel dat het aantal gebruikte verfijningsniveaus bij de lokale uniforme roosterverfijningsmethode constant is in de tijd, een algemeen Runge-Kutta schema gebruikt wordt voor het tijdstappen en de norm van de inverse van de Jacobiaan begrensd is, onafhankelijk van roosterafstand en tijdstap. Indien bij het creëren van de lokale sub-roosters dan aan de ongelijkheid (5.25) uit hoofdstuk 4 wordt voldaan, is de begrenzing van de lokale plaatsfout bij gebruik van de lokale uniforme roosterverfijningsmethode en bij gebruik van één uniform grid vergelijkbaar.

4. Bij de lokale uniforme roosterverfijningsmethode zal in het geval dat het DIRK schema uit referentie [4] van hoofdstuk 4 wordt gebruikt voor tijdsintegratie geen orde reductie optreden bij inwendige randen als de interpolatie die daar plaats vindt dezelfde of een hogere orde heeft dan de orde van de partiële differentiaalvergelijking.

5. Als de lokale uniforme roosterverfijningsmethode toegepast wordt op een stelsel gekoppelde partiële differentiaalvergelijkingen en er wordt een verfijningsstrategie gebruikt gebaseerd op foutschattingen, dan moet in de verfijningsstrategie onderscheid worden gemaakt tussen de componenten van de fout die behoren bij de verschillende partiële differentiaalvergelijkingen teneinde de wijze waarop deze

componenten de globale fout beinvloeden op een goede wijze tegen elkaar af te wegen.

Hoofdstuk 5 van dit proefschrift.

6. Het eerste orde split-schema, beschreven in [1], lost de lineare advectievergelijking

$$w_t=(aw)_x+(bw)_y, \qquad a=a(x,y), \quad b=b(x,y),$$

met tweede orde nauwkeurigheid op door de vergelijking als volgt te verstoren:

$$w_t=(\tilde{a}w)_x+(\tilde{b}w)_y, \quad \tilde{a}=a-\frac{1}{2}\tau(aa_x-ba_y), \quad \tilde{b}=b-\frac{1}{2}\tau(ab_x+bb_y).$$

[1] Willem Hundsdorfer and Ron Trompert, Method of lines and direct discretization - a comparison for linear advection, CWI rep. NM-R9314.

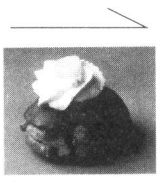
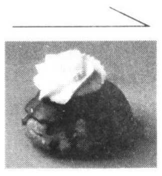
7. Goede badmintonspelers spelen op een schijnbaar kleinere baan dan slechte badmintonspelers.

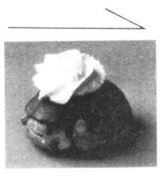
8. De landing van de eerste Spanjaarden in Mexico is vergelijkbaar met een landing van buitenaardse wezens op aarde in deze tijd.

R. van Zantwijk, De oorlog tegen de goden, Meulenhoff Amsterdam.

9. Inductie leidt tot vooroordelen, tenzij de inductie volledig is.

10. Het stelsel (on)gewone differentiaalvergelijkingen

$\frac{d}{dt}$  $=$ $($  $)$

wordt stijver naarmate de dimensie van de vector  toeneemt.